DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Material Design 3

Giacomo Checchini -1216347

Umberto Salviati - 1220994

Gabriel Taormina - 1219622

Sommario

INTRODUZIONE

Material Design è un sistema di linee guida, componenti e strumenti che supportano le migliori pratiche di progettazione dell'interfaccia utente.
In questo report abbiamo voluto illustrare nel dettaglio le sue principali caratteristiche e funzionalità.
Nella prima parte abbiamo introdotto il material design per poi concentrarci sulla presentazione del Material Design 3. Alla fine mostreremo la nostra applicazione che segue le regole dettate dal Material Design 3: "Diario M3".Abbiamo ritenuto interessante analizzare nel nostro report gli algoritmi che stanno alla base dei Dynamic Colors.

MATERIAL DESIGN

Material design è un linguaggio di design sviluppato da Google ispirato dagli elementi classici della stampa trasporto in un ambiente bidimensionale digitale, è supportato nativamente a partire da Android 5.0

Gli aspetti grafici compositivi creano e impostano il significato e gerarchia di tutti i contenuti disposti sullo schermo (fornendo una logica e un metodo secondo cui organizzare i contenuti) andando a mantenere l'interfaccia utente a prescindere dal dispositivo, mezzo e sistema operativo.

Il linguaggio è composto da una sintassi di effetti di profondità, transizioni, padding e tipografie, l'obiettivo è quello di fornire gli standard di progettazione specifici per lo sviluppo di applicazioni per dispositivi Android, iOS e web; unificando la UX degli utenti tra le diverse piattaforme ma allo stesso tempo essere un fondamento flessibile per l'innovazione.

Nel momento dell'interazione con l'utente il "Material" offre una esperienza più realistica, l'utilizzo di attributi tattili, visivi familiari al cervello rendono più comprensibile e intuitiva l'applicazione.

Rispetto ai design già utilizzati e presenti il Material Design offre palette di colori ampie, spunti visivi complessi, animati e interattivi nonché una maggior varietà di icone e forme. Tutti i vari elementi di tipografia, scale di colore, tabelle, spazi vuoti hanno l'obiettivo di creare una gerarchia avente un certo significato, focalizzando l'attenzione. La scelta di una determinata gamma di colori, dello spazio bianco intenzionale, le animazioni hanno lo scopo di migliorare la qualità dell'esperienza dell'utente rendendola più intuitiva. Una caratteristica fondamentale del material design è l'utilizzo delle superfici digitali o quantum paper, ogni elemento di un'interfaccia diventa una superficie reale toccabile; le superfici sono dallo spessore di 1 dpi

I principi alla base del Material Design sono i seguenti:

- Material Design si **ispira al mondo fisico** e alle sue trame, incluso il modo in cui riflettono la luce e proiettano le ombre. Le superfici dei materiali reinventano i mezzi di carta e inchiostro.
- Material Design è **guidato dai metodi di progettazione della stampa** - tipografia, griglie, spazio, scala, colore e immagini - per creare gerarchia, significato e messa a fuoco che immergono gli spettatori nell'esperienza.
- Il movimento **focalizza l'attenzione** e mantiene la continuità attraverso un feedback sottile e transizioni coerenti. Quando gli elementi appaiono sullo schermo, trasformano e riorganizzano l'ambiente con interazioni che generano nuove trasformazioni.

La scelta di una determinata gamma di colori, dello spazio bianco intenzionale, le animazioni hanno lo scopo di migliorare la qualità dell'esperienza dell'utente rendendola più intuitiva.

Material Design ha qualità tridimensionali che si riflettono nell'uso di superfici, elevazione e ombre.

I punti chiave del Material Design riguardano:

- Superfici
- Colori
- Testi
- Componenti
- Layout e movimento

Superfici

Nel mondo fisico, gli oggetti possono essere impilati o attaccati l'uno all'altro, ma non possono passare l'uno attraverso l'altro. Proiettano ombre e riflettono la luce. Material Design riflette queste qualità nel modo in cui le superfici vengono visualizzate e si spostano nell'interfaccia utente. Il modo in cui si muovono nelle tre dimensioni, è tale da ricordare il movimento nel mondo fisico. Le superfici dei materiali hanno caratteristiche e comportamenti costanti e immutabili. Il contenuto viene visualizzato in qualsiasi forma e colore, non aggiunge spessore ed è espresso senza essere ad un livello separato. É importante ricordare che il Material è “solido”, di conseguenza più elementi non possono occupare lo stesso punto nello spazio nello stesso momento, non è fluido o gassoso anche se può mostrare degli elementi che ricordano queste proprietà.

Le superfici dei materiali possono comportarsi in determinati modi: quelle rigide rimangono della stessa dimensione in tutte le interazioni, le estensibili possono crescere o restringersi lungo uno o più bordi fino a un limite di dimensioni, quindi comportarsi come superfici rigide, quelle “spostabili” rimangono delle stesse dimensioni durante le interazioni. Possono visualizzare contenuti aggiuntivi scorrendo all'interno dell'area, fino a raggiungere un limite di contenuti. Quando viene raggiunto questo limite, si comportano come superfici rigide in quella direzione di scorrimento. Le superfici possono rimanere in una posizione fissa sugli assi x e y, oppure possono essere spostate in qualsiasi direzione. Il movimento della superficie può essere limitato a un singolo asse, ad uno alla volta o ad entrambi
contemporaneamente.

Le superfici inoltre possono essere trasparenti, semitrasparenti o opache.

Gli **scrim** sono trattamenti temporanei che possono essere applicati alle superfici con lo scopo di rendere meno prominente il contenuto su una superficie. Aiutano a dirigere l'attenzione dell'utente su altre parti dello schermo.

I bordi aiutano ad esprimere la qualità tattile delle superfici. Mostrano dove finisce una superficie e inizia un'altra separando parti diverse di un'interfaccia utente in componenti identificabili. Per impostazione predefinita, le superfici utilizzano le ombre per indicare i bordi. Altri metodi possono essere utilizzati per indicare i bordi. Devono creare un contrasto sufficiente tra le superfici affinché siano viste come separate l'una dall'altra.

Elevazione:

L'elevazione in Material viene misurata come la distanza tra le superfici del materiale. La distanza dalla parte anteriore di una superficie materiale alla parte anteriore di un'altra viene misurata lungo l'asse z. Tutte le superfici e i componenti hanno valori di elevazione. Consente alle superfici di spostarsi davanti e dietro alle altre superfici riflettendo le relazioni spaziali. L'elevazione può essere rappresentata utilizzando ombre o altri segnali visivi, come riempimenti di superficie o opacità ed è anche usata per attirare e concentrare la visione di un elemento o notifica.

I componenti possono modificare l'elevazione in risposta all'input dell'utente o agli eventi di sistema ma una volta che l'input dell'utente è stato completato o annullato, il componente torna rapidamente alla sua altezza di riposo.

Per rappresentare correttamente l'elevazione, una superficie deve mostrare: bordi della superficie in contrasto con la superficie dall'ambiente circostante, sovrapporre ad altre superfici, distanza da altre superfici.

Le ombre possono esprimere il grado di elevazione tra le superfici in modi che altre tecniche non possono.

Tutti gli elementi hanno valori di default per l'elevazione. Alcuni componenti sono posizionati ad altitudini più elevate rispetto ad altri, stabilendo un ordine di elevazione coerente su tutti i componenti.

Ombre:

Le superfici proiettano delle ombre. In Material Design, le luci virtuali illuminano la UI. Le luci chiave creano ombre più nitide e direzionali, chiamate ombre chiave. La luce ambientale appare da tutte le angolazioni per creare ombre morbide e diffuse, chiamate ombre ambientali. In Android le ombre si verificano quando le sorgenti luminose sono bloccate dalle superfici del materiale in varie posizioni lungo l'asse z. Le ombre forniscono indicazioni sulla profondità, la direzione del movimento e i bordi della superficie. L'ombra di una superficie è determinata dalla sua elevazione e dalla relazione con le altre superfici.

Colori

I colori sono ideati per distinguere gli elementi e le superfici dell'interfaccia utente l'uno dall'altro cercando di mantenere armonia e garantendo la leggibilità e accessibilità al testo. Fondamentale è la gerarchia tra i vari colori poiché questa indica quali elementi sono interattivi e il loro livello di importanza, seguendo l'ottica che gli elementi più importanti sono quelli che più devono risaltare. Una semplice ed efficace linea guida sull'accostamento di colori è scegliere un colore primario e uno secondario legandoli da luci e giochi d'ombra.

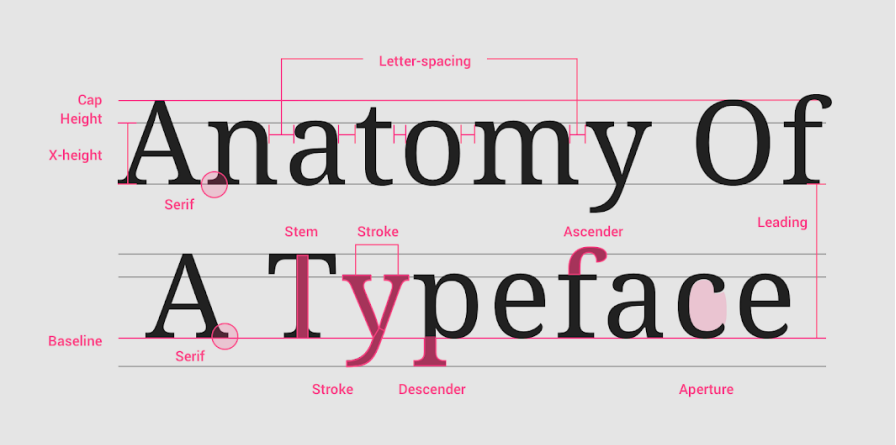
- Il **colore primario** è quello visualizzato più frequentemente nelle schermate e nei componenti dell'app
- Il **colore secondario** permette di accentuare e distinguere i vari elementi in più modi, è facoltativo e non va utilizzato in modo esagerato

I colori di testo o icone devono essere progettati in modo tale da essere chiari e leggibili rispetto ai colori dietro di essi. Questi colori sono chiamati colori "on", i valori predefiniti sono #FFFFFF e #000000

è presente un "generatore" di colori che genera una tavolozza per qualsiasi colore immesso, le varie tonalità, crome e luminosità sono regolate da un algoritmo.

Considerando la UI nel suo insieme, il colore deve creare una distinzione tra gli elementi, con sufficiente contrasto tra di loro ed essere applicato in modo mirato in quanto può trasmettere le relazioni tra gli elementi e i gradi della gerarchia. Infatti è il metodo più efficace per la realizzazione della gerarchia. Non bisogna eccedere con l'uso del colore, così facendo le aree che lo vanno ad utilizzare hanno maggior rilievo e attenzione. Material Design include già colori predefiniti per colori primari e secondari, varianti e colori dell'interfaccia utente aggiuntivi (sfondi, errori, tipografia) Il colore gioca un ruolo importante nella leggibilità del testo in particolare il testo nero è consigliato per l'uso su sfondi chiari e il bianco su sfondi scuri. Se si usano temi chiari e scuri assieme bisogna assicurarsi che il testo sia disponibile in un colore contrastante rispetto a ciascun tema.

Testi



Material Design include una vasta gamma di stili, contiene categorie di testo riutilizzabili, ciascuna con un'applicazione e un significato ben preciso. è presente un generatore di scale di caratteri, uno strumento grazie al quale qualsiasi carattere scelto viene ridimensionato automaticamente; alimentato da GoogleFont.

Sebbene ogni lettera sia unica, alcune forme sono condivise tra le lettere. Un carattere tipografico ovvero una raccolta di lettere rappresenta i modelli condivisi in una raccolta. I caratteri tipografici sono selezionati in base al loro stile, la leggibilità è più efficace quando seguono i principi fondamentali del design tipografico. I modelli sono:

La linea di base: è la linea invisibile all'utente su cui poggia il testo che specifica le distanze dagli elementi dell'interfaccia utente dalla linea di base. I valori di base sono indipendenti dal software, quindi funzionano in qualsiasi programma di progettazione e funzionano con la griglia.

Cap height: si riferisce all'altezza delle lettere maiuscole piatte misurata dalla linea di base. Le lettere maiuscole tonde e appuntite vengono regolate otticamente essendo disegnate con una leggera eccedenza sopra l'altezza del cap per ottenere l'effetto di essere della stessa dimensione. Ogni carattere tipografico ha un'altezza unica.

X-height: si riferisce all'altezza minuscola di un carattere e indica quanto sarà alto o corto ogni glifo.I caratteri con altezze 'X' alte hanno una migliore leggibilità con caratteri di piccole dimensioni, poiché lo spazio bianco all'interno di ogni lettera è più leggibile

Ascenders & descenders: gli ascenders sono un tratto verticale verso l'alto che si trova in alcune lettere minuscole che si estendono oltre cap height o la linea di base. I descenders sono il tratto verticale verso il basso in queste lettere. In alcuni casi può verificarsi una collisione tra questi tratti quando l'altezza della linea è troppo stretta.

Peso: il peso si riferisce allo spessore relativo del tratto di un font. Un carattere può avere molti pesi, il range tipico va da 4 a 6

I tipi sono:

Serif: una piccola forma o proiezione che appare all'inizio o alla fine di un tratto su una lettera.

Sans-Serif: assenza di un serif

Monospazio: caratteri con la stessa larghezza

Scrittura manuale: riproduco la scrittura manuale in stile corsivo,stampatello o con forti contrasti. Solitamente da H1 a H6

Display: categoria varia che comprende i tipi di grande dimensione e usati in punti specifici. Solitamente da H1 a H6

Componenti

I componenti sono elementi costitutivi interattivi per la creazione dell' interfaccia utente.

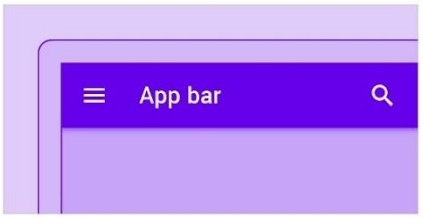
I componenti coprono una vasta gamma di esigenze di interfaccia, tra cui:

- **Visualizzazione**: posizionamento e organizzazione del contenuto utilizzando componenti come schede, elenchi e fogli.
- **Navigazione**: consente agli utenti di spostarsi all'interno del prodotto utilizzando componenti come cassetti e schede di navigazione.
- **Azioni**: consente agli utenti di eseguire attività utilizzando componenti come il pulsante di azione mobile.
- **Input:** consente agli utenti di inserire informazioni o effettuare selezioni utilizzando componenti come campi di testo, chip e controlli di selezione.
- **Notifiche**: avviso agli utenti di informazioni e messaggi chiave utilizzando componenti come snackbar, banner e finestre di dialogo.

I componenti principali usati nello sviluppo di applicazioni per Android sono i seguenti:

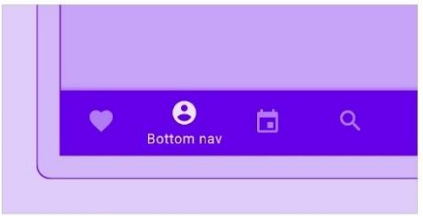
App bars: bottom

Una barra in basso mostra la navigazione e le azioni chiave nella parte inferiore degli schermi dei dispositivi mobili.



App bars: top

La barra dell'app in alto mostra informazioni e azioni relative alla schermata corrente.

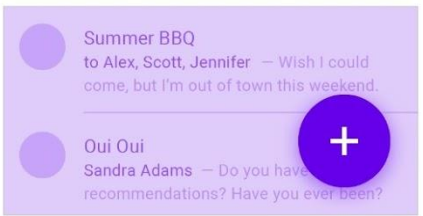


Bottom navigation

Le barre di navigazione in basso consentono lo spostamento tra le destinazioni principali in un'app.

Buttons

I pulsanti consentono agli utenti di intraprendere azioni e fare scelte con un solo tocco



Buttons: floating action button

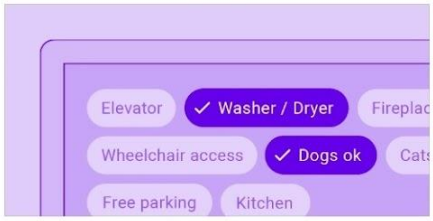
Un pulsante di azione mobile (FAB) rappresenta l'azione principale di uno schermo.

Cards

Le schede contengono contenuti e azioni su un singolo argomento.

Checkboxes

Le caselle di controllo consentono agli utenti di selezionare uno o più elementi da un set. Le caselle di controllo possono attivare o disattivare un'opzione.

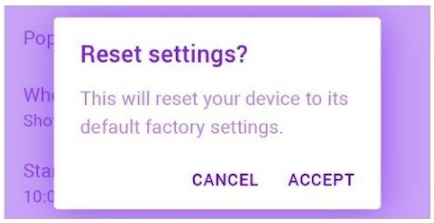


Chips

I chip sono elementi compatti che rappresentano un input, un attributo o un'azione

Date pickers

I selettori di date consentono agli utenti di selezionare una data o un intervallo di date.



Dialogs

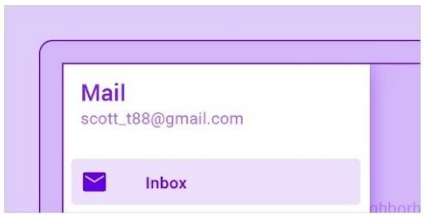
possono contenere informazioni critiche richiedere decisioni, coinvolgendo più attività

Dividers

Un divisore è una linea sottile che raggruppa il contenuto in elenchi e layout.

Menus

I menu mostrano un elenco di scelte sulle superfici temporanee.



Navigation drawer

I cassetti di navigazione forniscono l'accesso alle destinazioni nella tua app.

Navigation rail

I binari di navigazione forniscono movimenti ergonomici tra le destinazioni principali nelle app.

Progress indicators

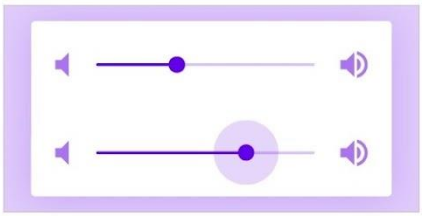
Gli indicatori di avanzamento esprimono un tempo di attesa non specificato o mostrano la durata di un processo

Radio buttons

I pulsanti di opzione consentono agli utenti di selezionare un'opzione da un set.

Sheets: bottom

I fogli inferiori sono superfici contenenti contenuto supplementare ancorate alla parte inferiore dello schermo

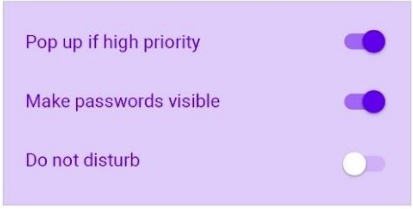


Sliders

I dispositivi di scorrimento consentono agli utenti di effettuare selezioni da un intervallo di valori.

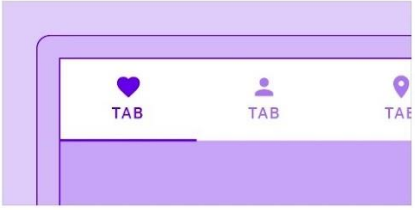
Snackbars

Gli snackbar forniscono brevi messaggi sui processi delle app nella parte inferiore dello schermo



Switches

Gli interruttori attivano o disattivano lo stato di un singolo elemento.

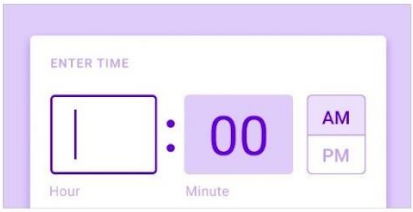


Tabs

Le schede organizzano il contenuto su diverse schermate, set di dati e altre interazioni.

Text fields

I campi di testo consentono agli utenti di inserire e modificare il testo.



Time pickers

I selettori orari aiutano gli utenti a selezionare e impostare un'ora specifica

Layout

Le varie aree di layout sono la base per le esperienze interattive dell'utente, i layout di Material Design utilizzano elementi e spaziature uniformi in modo da mantenere una coerenza tra le diverse piattaforme e dimensioni dello schermo. Le aree di layout possono raggruppare contenitori più piccoli al loro interno. In generale i layout devono utilizzare griglie, padding e linee chiave.

Andando a considerare un layout a schermo grande si hanno tre aree principali:

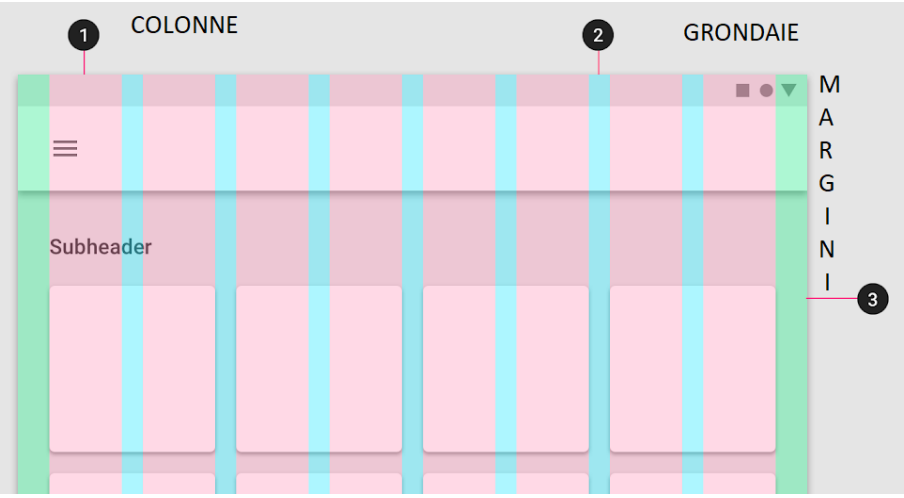
- **barra delle app:** utilizzata per visualizzare e raggruppare componenti per eseguire azioni primarie o su elementi presenti nel body
- **body**: zona in cui è visualizzata la maggior parte dei contenuti all'interno dell'app
- **zona di navigazione:** aiuta gli utenti a navigare tra le possibili destinazioni, contiene componenti di navigazione

Queste seppur in rapporti differenti sono elementi presenti anche in schermi più piccoli.
Per creare "ordine" in un layout il primo passo è il raggruppamento visivo: gli elementi con contenuto o funzioni simili all'interno di un layout vanno raggruppati e separati dagli altri elementi utilizzando spazi, divisori o testo. Segue il raggruppamento di "contenimento" dove si vanno a considerare elementi correlati da un contesto condiviso (es: immagine e la didascalia associata) attraverso la creazione di confini tra i vari gruppi.

Material Design suggerisce una lunghezza di riga ideale di 40-60 caratteri di testo per mantenere la leggibilità durante il ridimensionamento di elementi con testo, componenti invece sono dimensionati con incrementi di 8 dp per garantire un ritmo visivo coerente su ogni schermo.

Per visualizzare gli elementi in modo coerente su schermi con densità diversa le UI di Material Design utilizzano i "pixel indipendenti dalla densità", unità flessibili che ridimensionano per avere dimensioni uniformi su tutti i tipi di schermo (utile per portare un progetto su più piattaforme). Per garantire la coerenza tra i layout Material utilizza la griglia di layout reattiva che si adatta alla dimensione e orientamento dello schermo. Tale griglia è composta da tre elementi:

- **colonne:** la larghezza della colonna è definita con percentuali, questo consente ai contenuti di adattarsi a qualsiasi dimensione dello schermo. Il numero di colonne visualizzate nella griglia è determinato dall'intervallo del punto di interruzione. Un punto di interruzione può corrispondere a dispositivi differenti
- **"grondaie":** lo spazio tra le colonne che aiuta a separare il contenuto. Le larghezze sono valori fissi in ogni intervallo di punti di interruzione
- **margini**: lo spazio tra il contenuti e i bordi sinistro-destro dello schermo, per adattarsi alle dimensioni dello schermo la larghezza può cambiare in diversi punti di interruzione



Un ruolo importante è giocato dall'adattamento dei componenti che descrive le modifiche nella presentazione visiva o la sostituzione di un componente più adatto al caso d'uso e dimensione/ orientazione dello schermo.

Quando si ridimensiona un layout i componenti possono sia avere lunghezze fisse che adattive all'interno dell'intervallo dei vincoli di dimensione

Movimento

Un altro aspetto importante nel Material Design è quello del movimento tra i vari elementi poiché aiuta a rendere un'interfaccia espressiva e facile da utilizzare. Il movimento concentra l'attenzione su ciò che è importante.

Il sistema di movimento è composto da quattro modelli per la transizione tra i componenti o le visualizzazioni a schermo intero. I modelli sono progettati per aiutare gli utenti a navigare e comprendere un' app rafforzando le relazioni tra gli elementi dell'interfaccia utente, i modelli sono: trasformazione del contenitore, asse condiviso, sfumatura e dissolvenza.
Il modello di trasformazione del contenitore è progettato per le transizioni tra gli elementi dell'interfaccia utente che includono un contenitore. Questo modello crea una connessione visibile tra due elementi dell'interfaccia utente.

Il modello di asse condiviso viene utilizzato per le transizioni tra gli elementi dell'interfaccia utente che hanno una relazione spaziale o di navigazione. Questo modello utilizza una trasformazione condivisa sull'asse x, y o z.
Il modello di sfumatura viene utilizzato per le transizioni tra gli elementi dell'interfaccia utente che non hanno una forte relazione tra loro.
Il modello di dissolvenza viene utilizzato per gli elementi dell'interfaccia utente che entrano o escono entro i limiti dello schermo.

MATERIAL DESIGN 3

Nel Maggio del 2021, Google annunciò il nuovo e migliorato Material Design 3 definendolo come il “più espressivo e adattabile sistema di design del momento”.

Material Design 3 fu rilasciato prima per i dispositivi Pixel e poi successivamente per le altre tecnologie verso la fine del 2021.

Usando il nome di Material You questo consistente aggiornamento ha l’ambizione di aiutare tutti i designers.

“a creare design personali per ogni stile, accessibili per ogni esigenza, vivi e adattivi per ogni schermo”

Riportando così creatività ed entusiasmo per le app Android.

Un esempio importante è che, d’ora in avanti, le applicazioni ed i widget potranno essere personalizzati in base alla palette di colori generata a partire dallo sfondo degli utenti. Tutto ciò avviene in automatico senza il minimo sforzo degli utenti.

Un’altra potenzialità di Material You è che rende facile per gli sviluppatori creare app reattive e accessibili, offrendo più controllo per quanto riguarda i contrasti, dimensioni, spaziature ecc…

Dopo questa breve premessa è ora di addentrarci nella descrizione dettagliata di tutte le novità portate da Material You, che sono elencate in seguito:

- Colori Dinamici
- Rinnovamento dei componenti UI
- Design token

Dynamic Colors

I colori dinamici descrivono la potenzialità del sistema di colori di rispecchiare le preferenze e le impostazioni sui colori dell'utente

Material Design 3 è senza ombra di dubbio un aggiornamento audace. La novità più discussa ed apprezzata sono i cosiddetti "Dynamic Colors" o colori dinamici . Come già anticipato prima Material Design 3 ha una feature integrata di combinazione di colori, quest'ultimi vengono estratti dallo sfondo della schermata home dell'utente e applicati autonomamente a qualsiasi applicazione che accetti i colori dinamici. Permettendo un nuovo livello di personalizzazione delle app e differenziando, in maniera vistosa, la stessa applicazione a seconda dell'utente.
Pensando alla diversità degli sfondi dei vari utenti è immediato comprendere che la stessa app, distribuita in milioni di dispositivi diversi, sarà sempre differenziata in qualche modo rendendo così più originale e personalizzata l'esperienza dell'utente.

"Personal devices should feel personal" – Christian Robertson, creative lead of Material Design

L'abilitazione del colore dinamico nell'applicazione però deve la sua esistenza al sistema per la generazione e gestione dei colori di Material Design: il **color system.**
Procediamo ora ad analizzarlo nel dettaglio.

Color system

Il color system ideato da Material Design 3 garantisce una dinamicità agli **schemi di colore**, i quali si adatteranno sistematicamente ai dati forniti dall'utente. La logica delle relazioni e dei cambiamenti di tonalità e crominanza fornisce una base per un'applicazione dal colore dinamico.

Gli schemi di colore possono essere considerati come un gruppo coeso di tonalità relative, piuttosto che un insieme statico di valori costanti.

Colori chiave (key colors)

La base di uno schema di colori è un insieme di **cinque colori chiave**, da ognuno di essi verrà generata una palette differente con 13 (o più) variazioni di questo colore.

Questi colori chiave derivano tutti da un singolo colore, detto **source color.** Il source color può essere scelto in vari modi:

- Scelto a priori dallo sviluppatore.
- Scelto dagli algoritmi di Android 12 partendo dall'immagine di sfondo del dispositivo.
- Scelto dall'utente nella schermata di selezione dello sfondo.

In seguito andremo a spiegare dettagliatamente il processo di scelta del source color da parte di Android 12.

Ricapitolando, dal **source color** verranno derivati i 5 **key color** dai quali verranno create delle palette mantenendo costante la tinta e la crominanza del key color.

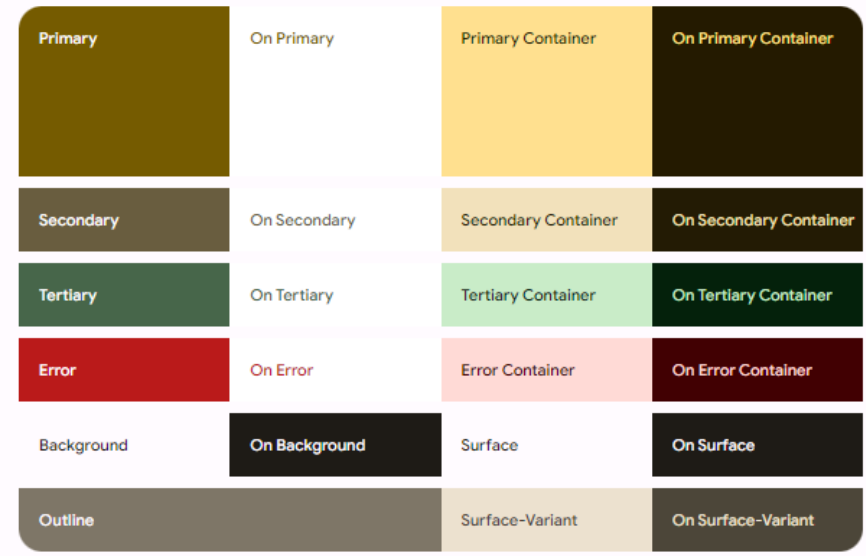
Alle tonalità di colore specifiche di ogni palette vengono assegnati dei **ruoli** ("color roles") da assumere nell'interfaccia utente.

Una volta stabiliti i key color sta ad un algoritmo di Material specificare lo spettro completo di colori necessari per esprimere gli stati di interazione, ad esempio gli errori, e garantire un contrasto accessibile.

Come vedremo in seguito ai 5 key colors si possono aggiungere dei colori personalizzati.

I colori chiave vengono divisi in **tre categorie**, queste categorie definiscono **i ruoli** che avranno i colori nella UI:

- ***Accent colors*** che comprende
  - **Primary Key color**: Si usa per i componenti chiave della UI come ad esempio FAB, pulsanti importanti, stati attivi e per colorare le superfici elevate.
  - **Secondary Key color:** Il suo uso è riservato ai componenti meno rilevanti della UI, ad esempio, i chips per il filtraggio
  - **Tertiary Key color:** Si usa per bilanciare i colori primari e secondari o per elevarne il contrasto, viene utilizzato inoltre per attirare maggiore attenzione su alcuni elementi.
- ***Neutral colors*** che comprende:
  - **Neutral Key Color:** vengono usati sia per gli sfondi e superfici sia per i testi ad alta importanza.
  - **Neutral Variant Key Color:** utilizzati invece per le variazioni di superficie ed icone e testi meno importanti.
- ***Additional colors*:**
  - **Error colors:** Questa categoria fornisce dei colori per i messaggi di errore.
  - **Product-specific custom colors:** Usati per permettere ai team di sviluppo più controllo e personalizzazione in un ambiente di colori dinamici autogenerati.

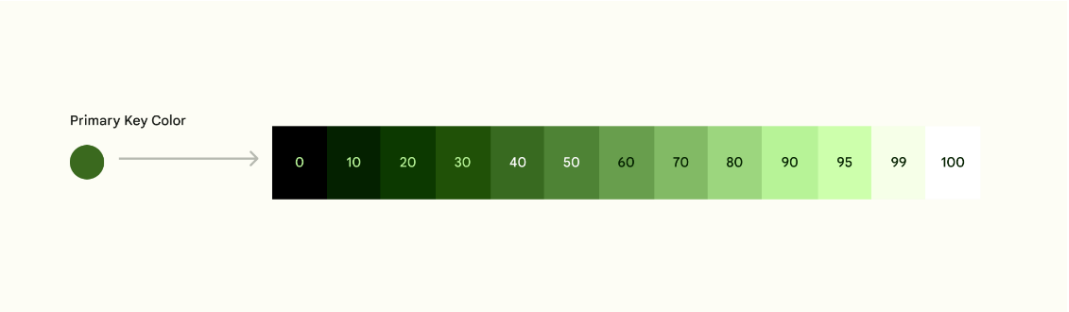


Oltre a questi ruoli ce ne sono altri di cui abbiamo omesso la descrizione, riportiamo ora la lista completa di tutti i ruoli disponibili in M3:

A cui vengono sommati **Inverse Surface, Inverse On Surface, Inverse Primary e Shadow** usati per visualizzare elementi di importanza minore nella UI.

Palette tonali (tonal palette)

Una palette tonale è composta da 13 variazioni di colori generate da un key color. Queste varianti vengono selezionate da un range di valori che va da 0 a 100, dove questo numero indica la quantità di luce presente nel key color. Al valore 0 sarà sempre associato il nero mentre al valore 100 il bianco.



Il valore della luminosità viene espresso insieme al ruolo del colore, ad esempio **primary40** è il primary key color con 40 come valore di luminosità.

Grazie allo strumento **Material Theme Builder** i designers possono creare la loro palette di colori e inserirla facilmente nell'app in sviluppo.

Questi schemi di colore personalizzati possono essere applicati a testi, icone, sfondi e persino illustrazioni, su tutte le pagine o aree selezionate. Ad esempio, puoi decidere di utilizzare i colori dinamici su una pagina del profilo utente ma non sulla home page.

Color schemes

Uno schema di colore (color scheme) è un gruppo di tonalità, prese dalla *tonal palette* e assegnate a ruoli specifici che vengono poi mappate nei componenti della UI.

Ad esempio da una *tonal palette* di un *primary key color* vengono scelti dei colori, per ruoli abbinati, come ad esempio un componente della UI e le sue scritte.

Un aspetto molto importante nella creazione di un color scheme è che ogni accoppiamento di colori non ha bisogno di essere testato nonostante l'alto grado di incertezza che comportano i dynamic colors.

Il sistema ideato da Material Design prevede infatti che ci siano delle relazioni prestabilite tra i colori di un tema che si basano sulla quantità di luce presente nel colore.

"By calculating luminance rather than hue we can define which tones combine successfully" – Bethany Fong, Design Lead of Material Design

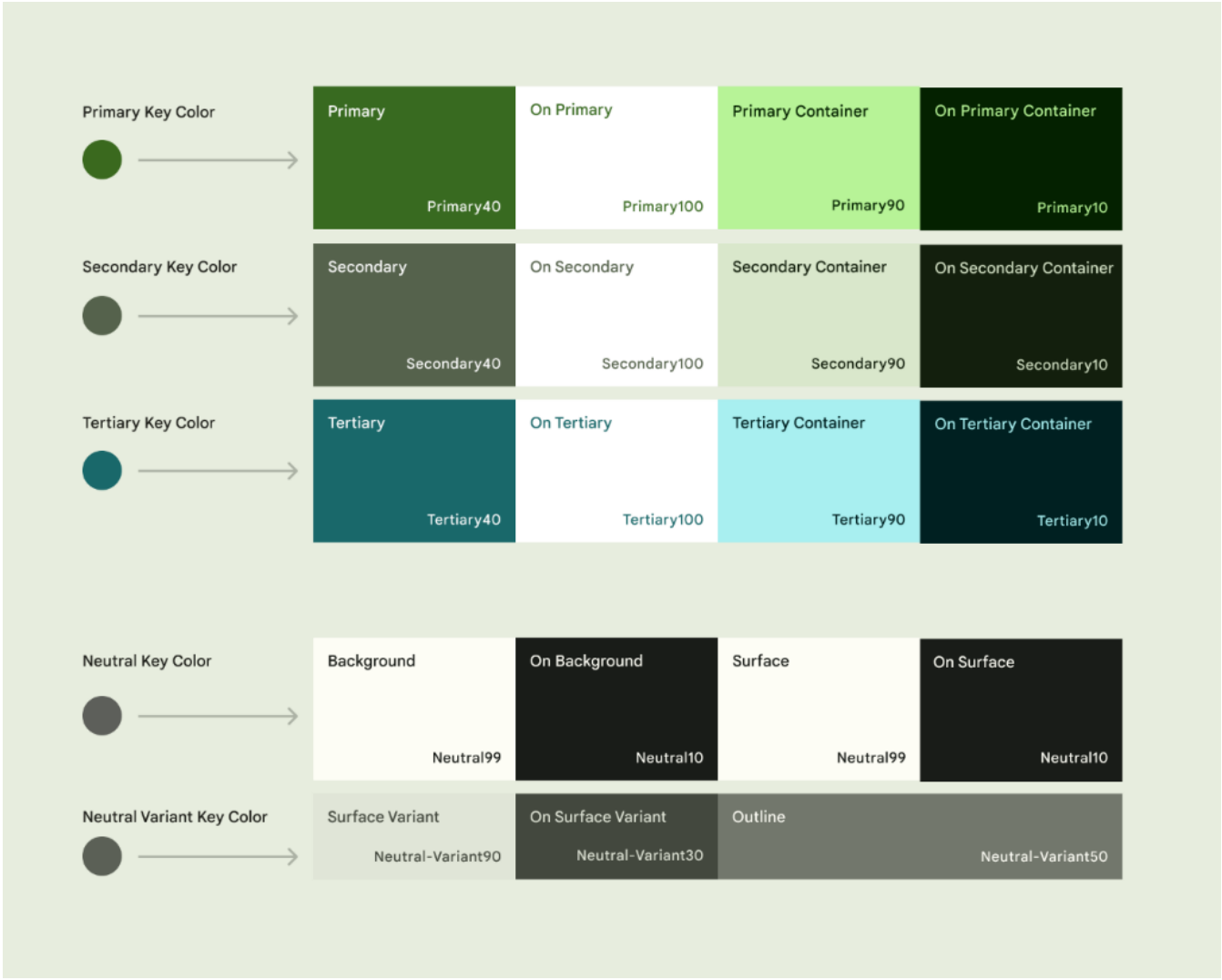
Quindi in un color scheme non ci saranno mai due colori ad alta (o bassa) luminosità combinati assieme, ciò renderebbe il loro contrasto poco efficiente e non funzionale per la UI.

Ogni *accent color* è fornito in un gruppo di 4 colori compatibili di luminosità differenti utilizzabili per gli abbinamenti. Questi quattro colori avranno i seguenti **ruoli**:

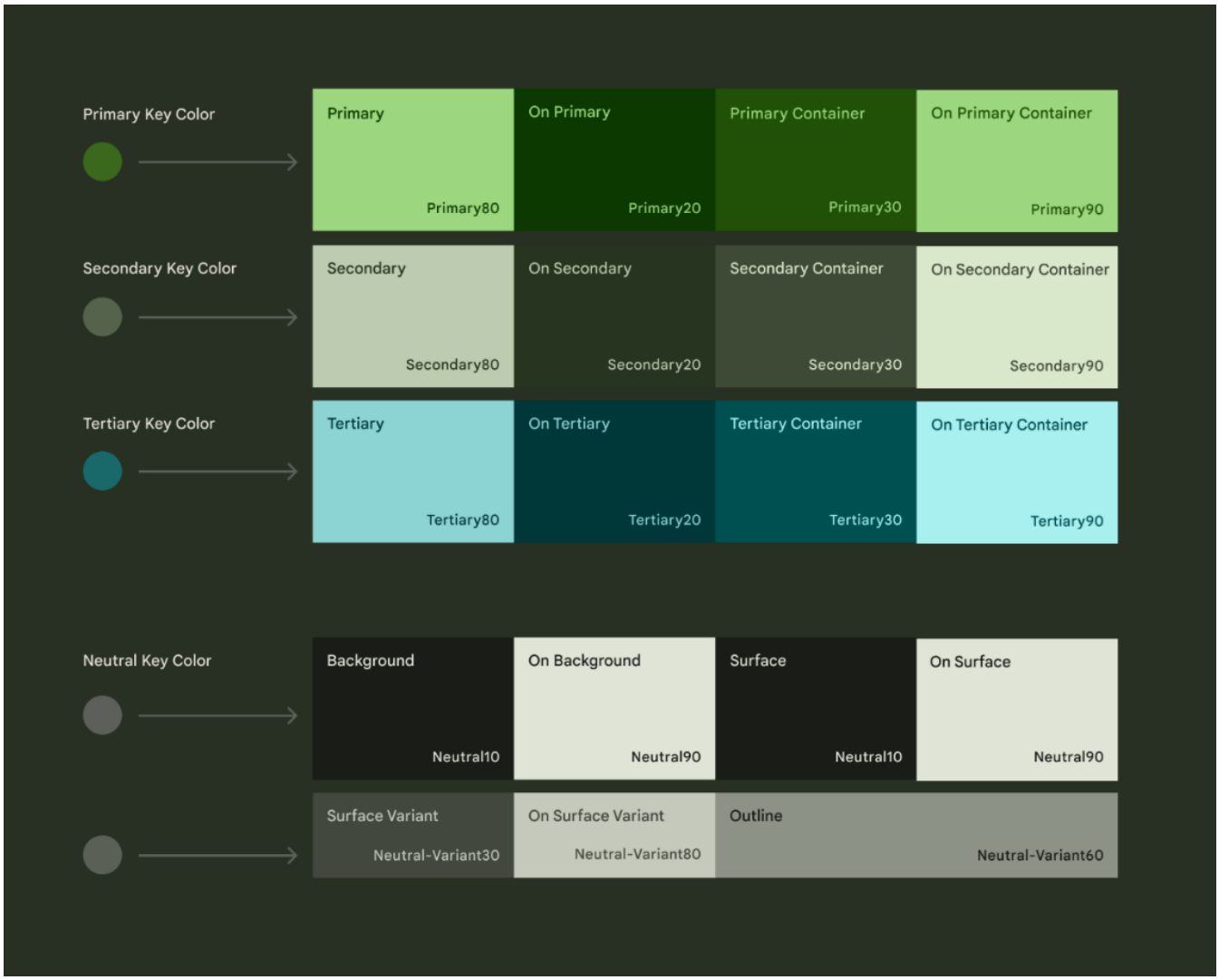
- **Primary:** è il colore base
- **On-primary:** applicato a tutto ciò che risiede sopra al colore primario (icone, testi , ecc)
- **Primary container:** Si applica agli elementi che hanno meno bisogno di risalto rispetto al primario
- **On-primary container:** applicato a tutto ciò che risiede sopra al *primary container* (icone, testi , ecc)

Questo pattern viene rispettato anche per i *secondary* e *tertiary colors*.

Vengono poi mappati anche i colori neutrali (*neutral colors*), verranno impiegati per superfici e sfondi ma anche testi ed icone ad alta rilevanza.



Alla generazione di un color scheme verranno mappati anche tutti i colori presenti nel tema scuro, essi verranno infatti generati dagli stessi **5 colori chiave** ma avranno, ovviamente, variazione di quantità di luce differenti rispetto a quelle presenti nel tema chiaro.



Un aspetto interessante da notare sono gli intervalli presenti tra i valori dei colori usati nel tema chiaro e quelli nel tema scuro.

Per il tema chiaro verranno selezionate i colori con valore 40-100-90-10 (seguendo l'ordine della figura) nel tema chiaro mentre nel tema scuro avremo sempre 80-20-30-90.

Prendendo in esame le tonalità del Primary Key Color si nota che il "container color" del tema chiaro è esattamente il "on container color" del tema scuro e che il primary color del tema chiaro verrà scurito di 10 valori e usato come "container color" del tema scuro. Tutto ciò evidenzia l'importanza delle *tonal palette* generata dal key color, esse infatti possono generare un'infinità di combinazioni di colori che però rimangono in armonia tra di esse e rispettano il tema comune del key color.

Custom colors

Dato quello che abbiamo visto finora i colori presenti in uno schema sono il risultato dei colori chiave e dei loro toni complementari, ma se un team di sviluppo volesse introdurre altri colori ad uno schema già definito?

I colori personalizzati subiscono lo stesso trattamento dello schema principale, quindi è possibile inserire un colore chiave personalizzato a cui vengono automaticamente assegnati un insieme di toni complementari.

In generale, i colori dello schema di base e i colori personalizzati comprendono formano l'intero schema. Chi sviluppa un' applicazione può scegliere da questo schema risultante quali colori applicare.

Dynamic Colors

Ora che abbiamo approfondito il sistema di colori che c'è dietro a M3 possiamo affrontare l'argomento dei colori dinamici.

Questa nuova feature di Android 12 si basa sul fatto che gli utenti possono creare schemi di colore personali in base ai loro sfondi oppure altre impostazioni di personalizzazione. Con Material Design 3 come base, i colori generati dagli utenti possono coesistere con i colori propri delle app, permettendo così un'ampia scelta di personalizzazione per l'esperienza visiva.

Vediamo adesso le diverse modalità per generare uno schema di colori:

User-generated scheme (UGS)

Uno schema generato dall'utente (traduzione italiana di UGS) viene estratto direttamente dai colori dello sfondo impostato nel dispositivo. L'interfaccia utente del sistema e tutte le app built-in (ad es calcolatrice, calendario, orologio) rifletteranno i colori presenti nello sfondo.

I colori che comporranno lo schema UGS vengono grazie alla seguente metodologia e poi applicati in tutto il dispositivo e app che accettano i colori dinamici.

In primo luogo, lo sfondo viene quantizzato, ovvero l'enorme quantità di colori presenti nell'immagine viene ristretta ad un insieme molto ridotto. Questo insieme risultante verrà poi sottoposto a degli algoritmi statistici che si occuperanno di filtrare questo insieme ed assegnare dei punteggi ai colori. Questi punteggi vengono assegnati in base alla porzione dell'immagine che rappresentano e alla loro adeguatezza nel creare dei temi formando così una classifica tra i colori.

In base alla classifica formata verrà individuato il **source color,** scelto dall'utente nel menu di personalizzazione dello sfondo oppure viene preso di default il primo classificato.

Dal source color verranno originati i **cinque key color** che genereranno a loro volta le loro **tonal palette.** Come già affermato in precedenza questi colori e le loro tonalità verranno associati a specifici ruoli e verranno mappati nei componenti della UI.

I colori destinati alla categoria "Error colors", vista in precedenza, non sono direttamente estratti dall'algoritmo ma vengono comunque salvati automaticamente.

Oltre agli sfondi un utente può selezionare i colori della UI anche grazie a schemi già predefiniti dal sistema, essi cambieranno l'aspetto dei componenti dell'interfaccia indipendentemente dal colore di sfondo.

M3 baseline color e tokens

Il “baseline color scheme” è un insieme di colori di default per creare temi (sia chiari che scuri) nelle applicazioni. Anche qui lo schema è strutturato in ruoli e relazioni tra diverse tonalità di colore, il che permette la perfetta interscambiabilità con un USG. Infatti il baseline color scheme utilizza gli stessi ruoli e mappature del USG.

Custom colors

Ricordiamo inoltre l’utilizzo dei custom colors che inserendosi in un qualsiasi schema di colori (UGS o baseline), permettono di definire ruoli di colore aggiuntivi specifici che sono essenziali per il significato di un’ app.

Design Tokens

Un'altra novità importante sono i design tokens, un nuovo strumento per migliorare la collaborazione consistente tra team di design e di sviluppo. Il potenziale di questo strumento è molto alto, soprattutto se si parla di progetti molto sofisticati e team di sviluppo molto ampi.

I tokens rappresentano i valori di uno stile dinamico, come ad esempio caratteri, colori o addirittura altri token, che possono essere aggiornati in tutto il progetto in una volta sola.

Struttura di un design token

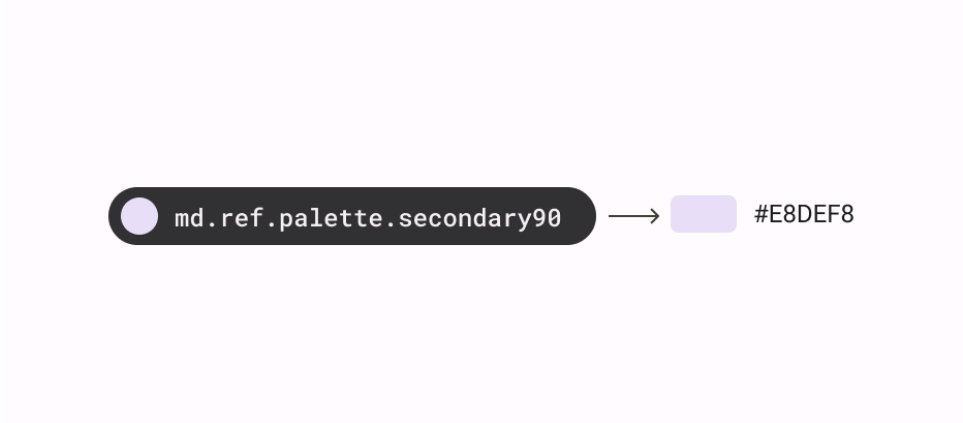
Un design token rappresenta una piccola, ripetuta decisione di design che compone lo stile visivo dell'intero sistema.

I design tokens rimpiazzano i valori statici, come ad esempio i valori esadecimali per i colori, con nomi auto esplicativi. Un token di Material Design è formato da due parti:

- Un **nome** che richiama il codice, come ad esempio **md.ref.palette.secondary90**
- Un **valore associato**, può essere il valore di un colore (#E8DEF8) come essere un tipo di carattere o una misura.

Il nome di un token è composto da più parti, separate da punti, che avanzano dalle informazioni più generali (md) alle più specifiche (secondary).

1. Tutti i nomi dei token iniziano con il nome del sistema di design, come ad esempio "md" che significa "Material Design"
2. Il nome prosegue con un'abbreviazione che descrive la tipologia del token:
    - **ref :** usato per i *reference tokens*
    - **sys :** indica i *system tokens*
    - **comp :** indica i *component tokens*
3. Il token finisce con un nome descrittivo che indica il ruolo del token nel sistema, ad esempio "palette.secondary90"



TIPOLOGIE DI TOKEN

Come anticipato prima esistono tre tipologie di token differenti: *reference tokens, system token* e *component tokens.* Analizziamo ora le loro caratteristiche.

Reference tokens:

Questi token includono tutte le opzioni stilistiche disponibili in un sistema di design. Solitamente si riferiscono ad un valore statico, come un codice esadecimale. I reference token non cambiano in base alla tipologia di utenti o al conteso del dispositivo.

Si usano principalmente per i colori e i caratteri, sia dimensioni che tipo. Possono addirittura riferirsi anche ad altri token della stessa tipologia.

System tokens:

Questa tipologia di token rappresenta le decisioni che sistematizzano il linguaggio di design per un tema o contesto specifico.

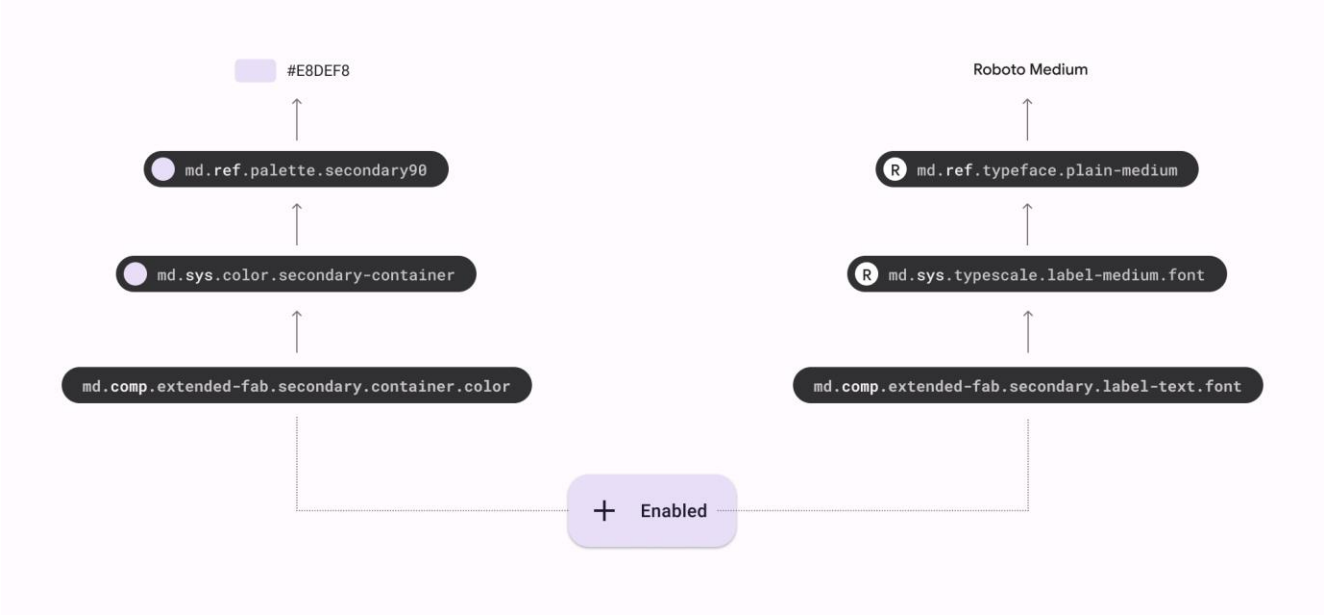
I *system tokens* definiscono lo scopo di un *reference token* nella UI. In altre parole, quando vengono applicati più temi in una app, un *system token* può puntare a diversi *reference token* in base al contesto, ad esempio un tema chiaro o scuro. Quando c'è la possibilità i system token devono puntare ad un reference token piuttosto che ad un valore statico

Component tokens:

I *component token* aggiungono un ulteriore livello nel sistema di design, rappresentano infatti gli elementi e i valori presenti in un componente della UI, come contenitori, testi delle etichette, icone e stati.

Come per i token di sistema, i component tokens devono puntare, quando possibile, a system o reference tokens evitando quindi l'associazione diretta con valori statici.

Questa tipologia di token è ancora in fase di sviluppo quindi il loro impiego deve essere ancora sfruttato a dovere.



IMPORTANZA DEI DESIGN TOKEN

I token permettono al sistema di design di avere una singola sorgente di verità. Permettono quindi la creazione di una raccolta e uno storico di tutte le decisioni stilistiche adottate e dei loro cambiamenti.

Questo utile strumento permette di gestire lo stile dell'applicazioni attraverso le fasi di design ed implementazione in maniera centralizzata. Non ci sarà quindi nessuna discrepanza nello stile dell'applicazione tra i file di design e il codice.
Ad esempio, se i modelli di un designer e il codice di un ingegnere fanno entrambi riferimento allo stesso token chiamato ad esempio "secondary container color", il designer e l'ingegnere sono certi che lo stesso colore verrà utilizzato da entrambe le parti. Questa coerenza rimane in atto anche quando il valore del colore assegnato a un token viene aggiornato.

In passato, gli stili di Material Design venivano comunicati tramite guidelines, file di design, strumenti e librerie per piattaforme specifiche.

Ora invece usando i token gli aggiornamenti degli stili saranno sempre effettuati in maniera consistente in tutto il progetto o addirittura in un insieme di applicazioni. È proprio qui che risiede la forza dei design tokens.

Poiché i token sono riutilizzabili e finalizzati ad uno scopo ben preciso, essi possono definire aggiornamenti a livello di sistema per temi e contesti da utilizzare.

Ad esempio, i token possono essere utilizzati per applicare sistematicamente, ad ogni aggiornamento, una combinazione di colori ad alto contrasto per una migliore visibilità o per modificare la scala dei caratteri per rendere leggibile un testo di piccole dimensioni anche su un televisore.

I design tokens diventano particolarmente utili ed efficienti in queste situazioni:

- Si ha necessità di aggiornare spesso il design di un prodotto o crearne uno nuovo
- Il sistema di design è applicato in diverse piattaforme e in più prodotti.
- Per sfruttare al meglio le nuove funzionalità di Material Design 3, come i colori dinamici.

Invece il loro utilizzo non trova particolare efficacia, quando:

- L'applicazione utilizza valori statici che non verranno cambiati per anni
- Il prodotto non ha un sistema di design predefinito

Design tokens nelle guidelines di Material Design

Nell guidelines fornite da Material Design si trovano delle tabelle che contengono liste di tokens per i colori e la tipografia. Questi tokens rappresentano i valori di default degli elementi a cui si riferiscono.

Le tabelle mappano la relazione tra un ruolo, il suo token di sistema, il reference token e il suo valore di default.

Ogni componente nelle guidelines ha una propria sezione, chiamata Specs, che elenca tutti i possibili design token e i valori di default a loro associati.

UI Components

Con Material You i componenti dell'interfaccia utente hanno avuto un aggiornamento dello stile adottando un look ancora più semplice e immediato.

Material 3 introduce un design più tondeggiante, con ombre più leggere e spazi bianchi diffusi, dedito all'implementazione dei colori dinamici.

Vediamo ora in rassegna le novità, componente per componente.

PULSANTI

Material include ora nove tipi di pulsanti(elevated button, filled button, filled tonal button, outlined button, text button, icon button, segmented button, fab, extended fab) rivisitati nella stile e nella forma

- colore: nuove mappature dei colori e compatibilità con il colore dinamico. Icone ed etichette ora condividono lo stesso colore.
- icone: la dimensione standard per le icone iniziali e finali è ora di **18 dp**
- forma: raggio dell'angolo completamente arrotondato, altezza più alta e nuova larghezza minima
- tipografia: il testo del pulsante è in maiuscolo
- tipi: tre nuovi tipi di pulsanti - pulsante elevato, pulsante riempito e pulsante tonale pieno - sostituiscono quello che in precedenza era noto come pulsante contenuto.

Con i pulsanti Material suggerisce di rispettare una gerarchia: ogni schermata dovrebbe contenere un singolo pulsante prominente per l'azione principale. Questo pulsante è detto ad alta enfasi. Ha lo scopo di attirare la massima attenzione. La disposizione degli elementi sullo schermo dovrebbe comunicare chiaramente che altri pulsanti sono meno importanti. Gli altri pulsanti devono essere disposti e definiti in modo da assicurarsi che lo stato disponibile di un pulsante non assomigli allo stato disabilitato di un altro. Il livello di "enfasi" di un pulsante aiuta a determinare l'aspetto, la tipografia e la posizione. È inoltre sconsigliato posizionare un pulsante sotto un altro pulsante se c'è spazio per posizionarli uno accanto all'altro.

FAB

- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- forma: stile boxier con raggio d'angolo più piccolo
- tipi: Nuova variazione large FAB

APP BAR INFERIORE

- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- altitudine: Nessuna ombra
- layout: l'altezza del contenitore è più alta e il FAB è ora contenuto nel contenitore della barra dell'app

CARDS

- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- elevazione: elevazione inferiore e nessuna ombra per impostazione predefinita
- tipi: tre tipi di carte ufficiali: **elevated, filled** e **outlined**

CHIPS

- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- forma: rettangolo arrotondato
- tipi: i gettoni azione sono stati separati in gettoni assistenza e gettoni suggerimenti. I chip Choice sono ora un sottoinsieme dei chip filtro

DIALOG

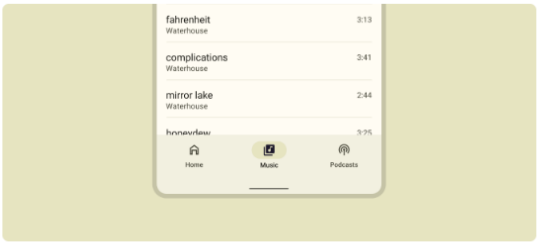
- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- layout: maggiore riempimento per tenere conto del raggio dell'angolo e delle dimensioni del titolo aumentati
- posizione: opzione per il posizionamento personalizzato della finestra di dialogo di base
- forma: raggio d'angolo aumentato
- tipografia: titolo più grande e più scuro

MENU

- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- tipi: il menu a discesa e il menu a discesa esposto ora sono entrambi indicati come menu, poiché differiscono solo per l'elemento che apre la superficie del menu

BARRA DI NAVIGAZIONE

- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- elevazione: Nessuna ombra
- layout: l'altezza del contenitore è più alta
- stati: La destinazione attiva può essere indicata con una forma a pillola in un colore contrastante
- nome: la “bottom bar” è stata rinominata in “navigation bar” ("barra di navigazione")



Navigation bar

CASSETTO DI NAVIGAZIONE - “DRAWER”

- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- forma: Angoli arrotondati sul bordo terminale del cassetto
- stati: colore e forma aggiornati per indicare lo stato selezionato

NAVIGAZIONE A ROTAIA - “RAIL”

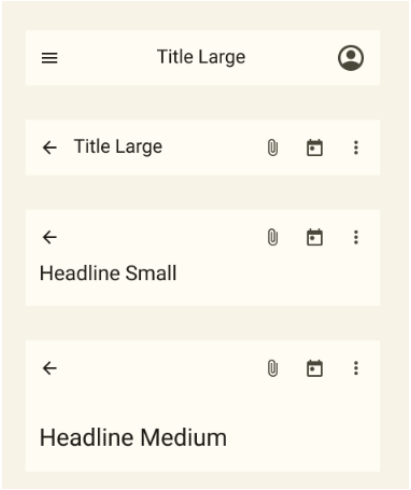
- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- stati: La destinazione attiva può essere indicata con una forma a pillola in un colore contrastante

SWITCH

- accessibilità: la presentazione visiva è più accessibile
- colore: le nuove mappature dei colori soddisfano i requisiti di non contrasto del testo oltre alla compatibilità con il colore dinamico
- icone: possibilità di avere un'icona opzionale all'interno del pollice dell'interruttore
- layout: la pista è più alta e più larga

CAMPI DI TESTO

- colore: nuove mappature dei colori e compatibilità con il colore dinamico

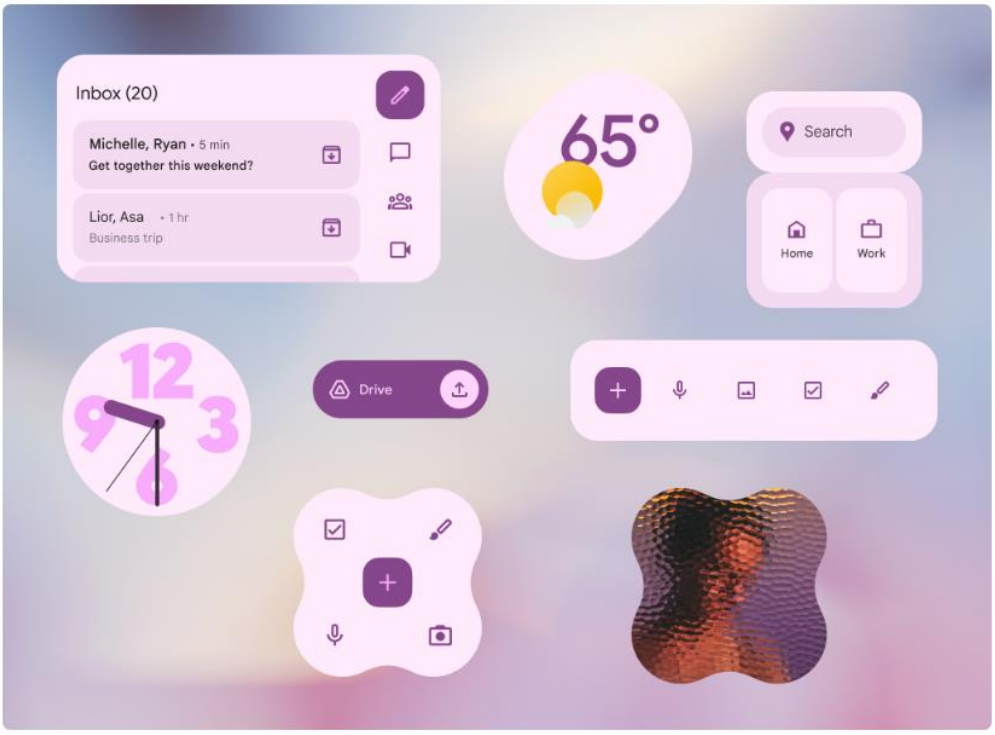


APP BAR SUPERIORE

- colore: nuove mappature dei colori e compatibilità con il colore dinamico
- elevazione: nessuna ombra discendente, invece un riempimento colore crea separazione dal contenuto
- tipografia: testo predefinito più grande
- layout: altezza predefinita maggiore
- tipi: ora ci sono quattro tipi di barra delle app in alto: allineata al centro, piccola, media e grande

WIDGETS

- Colore dinamico
- Comportamento con stato utilizzando caselle di controllo, interruttori e pulsanti di opzione
- Angoli arrotondati
- Attributi di dimensione perfezionati e maggiore flessibilità grazie a vincoli di dimensionamento aggiuntivi e layout reattivi



Material Components per Android (MDC)

Per le app Android, Material Components for Android (MDC Android) unisce design e ingegneria con una libreria di componenti per creare coerenza nell'applicazione. Con l'evolversi del sistema di Material Design, questi componenti vengono aggiornati per garantire un'implementazione coerente e perfetta per i pixel e l'aderenza agli standard di sviluppo front-end di Google. MDC è disponibile anche per Web, iOS e Flutter

I MDC è disponibile tramite la Maven repository di Google, per utilizzarli bisogna nel file build.gradle dell'applicazione verificare la presenza nella sezione repositories della repository di Google (google()) e aggiungere la libreria alle dipendenze. Per utilizzare i nuovi temi e stili di Material3 la versione della libreria deve essere 1.5.0 o successiva.

È possibile utilizzando RefactorToAndroidX aggiornare le dipendenze e il codice per utilizzare le librerie di androidx e material se l'applicazione dipende dalla "originale" Design Support Library.

I temi giocano un ruolo fondamentale in Material, nella versione 3 i temi permettono di avere un inflater di view personalizzato che sostituisce i componenti predefiniti con la controparte Material Gli ultimi stili dei componenti e i loro equivalenti sono i seguenti:

Material3	MaterialComponents
Theme.Material3.Light	Theme.MaterialComponents.Light
Theme.Material3.Light.NoActionBar	Theme.MaterialComponents.Light.NoActionBar
Theme.Material3.Dark	Theme.MaterialComponents
Theme.Material3.Dark.NoActionBar	Theme.MaterialComponents.NoActionBar
Theme.Material3.DayNight	Theme.MaterialComponents.DayNight
Theme.Material3.DayNight.NoActionBar	Theme.MaterialComponents.DayNight.NoActionBar
N/A	Theme.MaterialComponents.Light.DarkActionBar
N/A	Theme.MaterialComponents.DayNight.DarkActionBar

Per aggiornare il tema dell'applicazione ed ereditarlo da uno di quelli elencati bisogna inserire la seguente definizione

```
<style name="Theme.MyApp" parent="Theme.Material3.Light.NoActionBar">
    <!-- ... -->
</style>
```

È possibile testare in modo incrementale i nuovi componenti Material senza modificare il tema dell'app.

Ciò permette di mantenere lo stesso aspetto e comportamenti dei layout esistenti ma introducendo al contempo nuovi componenti nel layout uno alla volta. Tuttavia è necessario aggiungere dei nuovi attributi al tema dell'app per evitare errori di ThemeEnforcement.

COLORI
Il sistema di temi cromatici di Material3 può essere utilizzato per creare una combinazione di colori unici, utilizza un approccio organizzato per applicare i colori all'interfaccia utente. I componenti del materiale utilizzano i colori del tema e le relative variazioni per definire lo stile di sfondi, testo e altro.

Tutti i componenti di Material 3 utilizzano uno stile Widget.Material3 e questi stili fanno riferimento agli attributi di colore del tema. È possibile personalizzare gli attributi di colore sovrascrivendoli nel tema. Sono forniti tre gruppi di colori: primario, secondario, terziario; ciascuno con 4-5 ruoli colore personalizzabili.
Modificando questi attributi di colore, è facile cambiare gli stili di tutti i componenti Material che utilizzano il tema.

Material3 utilizza sovrapposizioni di prospetti di colore primario per presentare una gerarchia visiva con prospetti diversi sia nei temi chiari che in quelli scuri. I temi Material3 lo abilitano in modo predefinito con l'impostazione:

?attr/elevationOverlayColor su ?attr/colorPrimary.

Le elevazioni verranno applicate ai colori della superficie per creare variazioni tonali. All'interno della tavolozza dei colori di Material3 ci sono cinque variazioni tonali di superficie predefinite (Surface1-5) che vengono utilizzate come colori di superficie predefiniti applicando diverse elevazioni per diversi componenti.

Questi colori tonali di superficie non sono tuttavia implementati come risorse di colore ma i loro valori effettivi vengono calcolati al volo con:

?attr/elevationOverlayColor

Material3 utilizza una tonalità di viola per i colori predefiniti se i colori dinamici non sono disponibili, qualora si volessero usare colori diversi è possibile definire colori personalizzati per il proprio tema. È importante però mantenere lo stesso livello di luce quando si vanno a creare colori personalizzati per non violare i requisiti di contrasto. Quando si vanno a definire i colori personalizzati è consigliato utilizzare nomi rilevanti per il valore RGB e non lo stesso nome dello “slot” di colore

```
1  <resources>
2    <color name="brand_green">...</color>
3    <color name="brand_red">...</color>
4  </resources>
```

Se si vuole modificare il colore di una sola istanza di un componente senza però andare a modificare tutti gli attributi a livello di tema è possibile creare un nuovo stile per il componente, che si estende da uno stile Widget.Material3
Se invece si vogliono modificare gli stili predefiniti per tutte le istanze di quel componente bisogna andare a modificare l’attributo NomeComponenteStyle nel suo tema

```
<!--Singola istanza--> ->
<style name="Widget.MyApp.Button" parent="Widget.Material3.Button">
    <item name="backgroundTint">?attr/colorSecondary</item>
</style>

<!--Tutte le istanze-->
<style name="Theme.MyApp" parent="Theme.Material3.Dark.NoActionBar">
    ...
    <item name="materialButtonStyle">@style/Widget.MyApp.Button</item>
    ...
</style>
```

Per garantire la coerenza/coesione visiva in qualsiasi tema con colori dinamici abilitati e colori personalizzati si utilizza l'armonizzazione del colore, fornita dal pacchetto com.google.android.material.color andando a chiamare il metodo nella activity,fragment o view corrispondente

```
int harmonizedColor = MaterialColors.harmonizeWithPrimary(context, colorToHarmonize);
```

I metodi sono:

Static Factory Methods	Description
HarmonizedColorAttributes.create(int[] attributes)	Fornisce una matrice int di attributi per l'armonizzazione
HarmonizedColorAttributes.create(int[] attributes, int themeOverlay)	Fornisce un themeOverlay, insieme alla matrice int di attributi dall'overlay del tema per l'armonizzazione.
HarmonizedColorAttributes.createMaterialDefaults()	Fornisce un'implementazione predefinita di HarmonizedColorAttributes, con l'armonizzazione dei colori di errore.

Nel primo metodo l'ID della risorsa e il valore dell'attributo sono risolti in fase di esecuzione.
Il secondo metodo è consigliato invece per evitare di sovrascrivere senza volere le risorse di colore, in quanto verranno armonizzate le risorse di colore indicate dagli attributi colore dopo l'applicazione della sovrapposizione del tema.
È possibile utilizzare l'armonizzazione delle risorse di colore separatamente dai colori dinamici, ma il caso d'uso "generale" è dopo che sono stati applicati i colori dinamici, per garantire la coesione visiva per i colori riservati.

TEMA SCURO

Un tema scuro è generalmente costituito da colori di sfondo scuri e colori di primo piano chiari per elementi come testo e iconografia.
I vantaggi di un tema scuro includono: migliore risparmio energetico della batteria per i dispositivi con schermi OLED, affaticamento degli occhi ridotto e una migliore visibilità in ambienti scarsamente illuminati.
A partire da Android Q gli utenti ora possono passare il proprio dispositivo a un tema scuro tramite una nuova impostazione di sistema, che si applica sia all'interfaccia utente del sistema Android che alle app in esecuzione sul dispositivo (prima di poter utilizzare la funzionalità del tema materiale scuro, è necessario aggiungere una dipendenza alla libreria Material Components per Android)
Esempi possibili temi scuri e uno chiaro per l'applicazione, dove è consigliato definire in due directory diverse i temi chiari da quelli scuri

```
<style name="Theme.MyApp" parent="Theme.Material3.DayNight">
    <!-- ... -->
</style>

<style name="Theme.MyApp" parent="Theme.Material3.Light">
    <!-- ... -->
</style>

<style name="Theme.MyApp" parent="Theme.Material3.Dark">
    <!-- ... -->
</style>
```

Il tema Theme.Material3.Dark è un tema scuro statico mentre Theme.Material3.DayNight è un tema più dinamico che aiuta a passare facilmente dal tema chiaro a quello scuro. Se si utilizza un tema DayNight è possibile definire un tema dell'app che faccia riferimento alle risorse colore che possono essere sovrascritte nella directory values-night.
I temi Material Dark utilizzano il Material Color System per fornire valori di temi scuri predefiniti per i colori neutri della tavolozza come android:colorBackground e colorSurface. Lo sfondo del tema Materiale scuro di base e i colori della superficie sono grigio scuro anziché nero, il che aumenta la visibilità delle ombre e riduce anche l'affaticamento degli occhi per il testo chiaro.
Le ombre sono meno efficaci in un'app che utilizza un tema scuro, per compensare a ciò le superfici Material diventano più chiare e colorate a quote più elevate, quando sono più vicine alla fonte di luce implicita.
Ciò si ottiene tramite sovrapposizioni di prospetto, che sono sovrapposizioni semitrasparenti che sono concettualmente posizionate sopra il colore della superficie. A partire dai temi Theme.Material3.*, le sovrapposizioni di prospetto sono abilitate anche per i temi di luce.
Per supportare le sovrapposizioni di prospetto in una view personalizzata bisogna creare un MaterialShapeDrawable con il supporto di sovrapposizione abilitato tramite MaterialShapeDrawable#createWithElevationOverlay e impostarlo come sfondo della view, poi sovrascrivere il metodo View#setElevation e inoltrare l'elevazione passata al metodo setElevation dello sfondo MaterialShapeDrawable.

FORME e TEMA

La libreria Material Components offre una libreria di forme che può essere utilizzata per creare forme non standard utilizzando MaterialShapeDrawable, un Drawable che può disegnare forme personalizzate tenendo conto di ombre, elevazione, scala e colore. Oltre alla libreria delle forme, la libreria dei componenti dei materiali fornisce un meccanismo con cui personalizzare facilmente le forme dei componenti a livello di tema. Il tema Shape offre un'ulteriore possibilità di personalizzare l'aspetto grafico dell'applicazione.
MaterialShapeDrawable inizia con un path generato da ShapeAppearanceModel. Un ShapeAppearanceModel è composto da CornerTreatments e EdgeTreatments che si combinano per creare un path di forma personalizzato che in genere è passato a un costruttore di MaterialShapeDrawable (la libreria di forme fornisce alcune sottoclassi di CornerTreatments e EdgeTreatments per semplificare la creazione di nuove forme). I componenti supportati da MaterialShapeDrawables possono utilizzare i temi in un'applicazione.
Le forme dei componenti sono supportate da "apparenze di forma" ovvero riferimenti di stile che definiscono gli aspetti della forma. ShapeAppearanceModel usa lo stile shapeAppearance e crea trattamenti di angoli e bordi dai valori di shapeAppearance.
La libreria Material Components supporta forme tematiche a livello di applicazione. Per applicare le forme del tema all'applicazione bisogna specificare gli attributi del tema della forma nel tema. Ciò consentirà ai componenti che supportano il tema delle forme di leggere i valori personalizzati e di modificare le loro forme di conseguenza. Gli attributi del tema della forma devono puntare a stili shapeAppearance personalizzati che definiscono sia cornerSize che cornerFamily.

Un esempio di stile di shapeAppearance:

```
<style name="ShapeAppearance.MyAppMaterial.SmallComponent" parent="ShapeAppearance.Material3.SmallComponent">
  <item name="cornerFamily">cut</item>
  <item name="cornerSize">4dp</item>
</style>

<style name="ShapeAppearance.MyAppMaterial.MediumComponent" parent="ShapeAppearance.Material3.MediumComponent"
  <item name="cornerFamily">cut</item>
  <item name="cornerSize">8dp</item>
</style>

<style name="ShapeAppearance.MyAppMaterial.LargeComponent" parent="ShapeAppearance.Material3.LargeComponent">
  <item name="cornerFamily">rounded</item>
  <item name="cornerSize">4dp</item>
</style>
```

È possibile modificare la forma di un componente nell'intera applicazione definendo uno shapeAppearanceOverlay personalizzato nello stile del componente e anche modificare la forma di un singolo componente definendo uno stile shapeAppearanceOverlay personalizzato con solo l'attributo che si desidera sovrapporre sopra lo shapeAppearance esistente.

TESTO

È possibile utilizzare le scale di testo per personalizzare l'aspetto del testo nei componenti Material.
I componenti inclusi nella Material Design Library fanno riferimento a questi attributi di testo, consentendo di modificare facilmente l'aspetto del testo nell'intera applicazione. È possibile cambiare l'aspetto di qualsiasi stile di testo creando un nuovo stile e impostandolo nel tema

Un esempio:

```
<style name="TextAppearance.MyAppMaterial.DisplaySmall" parent="TextAppearance.Material3.DisplaySmall">
  ...
  <item name="fontFamily">@font/custom_font</item>
  <item name="android:textStyle">normal</item>
  <item name="android:textAllCaps">false</item>
  <item name="android:textSize">64sp</item>
  <item name="android:letterSpacing">0</item>
  ...
</style>

<style name="Theme.MyAppMaterial" parent="Theme.Material3.DayNight.NoActionBar">
  ...
  <item name="textAppearanceDisplaySmall">@style/TextAppearance.MyApp.DisplaySmall</item>
  ...
</style>
```

LAYOUT

Le demo di layout nel catalogo MDC sono esempi di layout adattivi in cui i componenti e le viste cambiano a seconda della configurazione del dispositivo, come le dimensioni dello schermo, l'orientamento e/o la presenza di una piega fisica.
Ogni demo ha una mainActivity che mostra il componente di navigazione appropriato in base alle dimensioni dello schermo, mostra il fragment principale e comunica con quella classe di fragment. Vengono utilizzati ConstraintLayout e ConstraintSet per consentire ai layout di adattarsi a configurazioni multiple di schermi e dispositivi e la libreria WindowManager per acquisire stati pieghevoli specifici. Tutte le demo mostrano diversi componenti di navigazione in base alle dimensioni dello schermo: gli schermi piccoli hanno una navigazione in basso, gli schermi medi hanno una barra di navigazione e gli schermi grandi hanno un cassetto di navigazione standard. Gli schermi medi visualizzano anche un riquadro di navigazione modale se si fa clic sul pulsante dell'intestazione di navigazione.

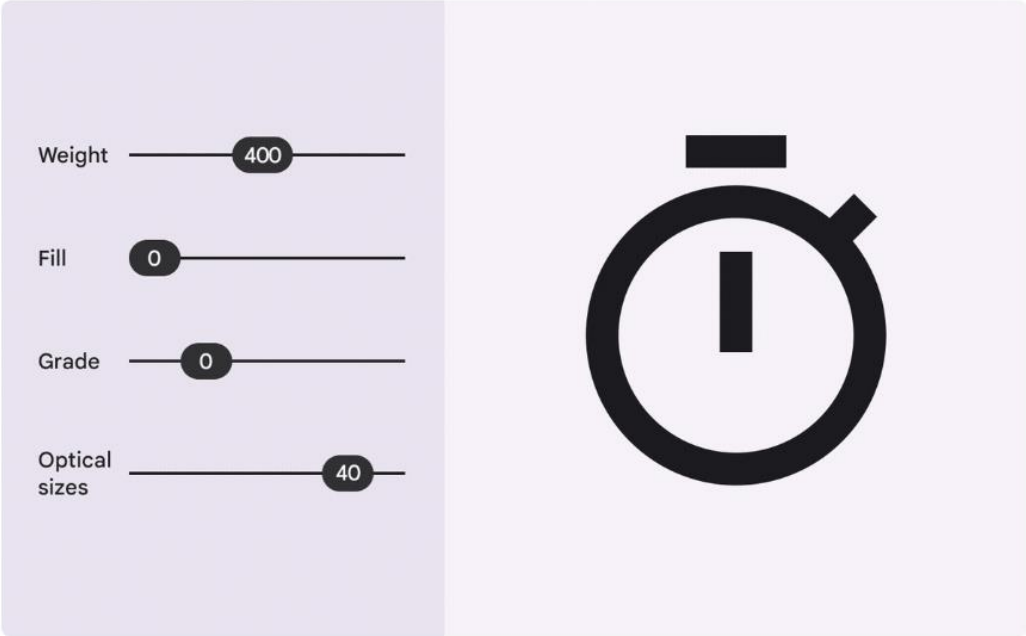
Icone

Material You ha portato anche un aggiornamento per quanto riguarda le icone in un' applicazione. Grazie ad esse è possibile rappresentare azioni comuni nelle nostre interfacce.

Material Symbols è un nuovo insieme di icone dove ognuna di esse supporta tre stili differenti:

1. **Outlined**
2. **Rounded**
3. **Sharp**

Inoltre Material Symbols permette di personalizzare le icone facendo variare 4 attributi: **Weight, Fill, Grade** e **Optical sizes.**



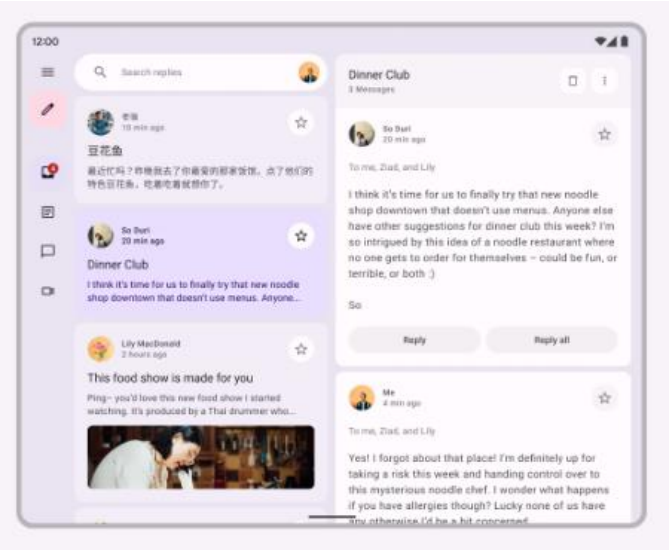
Dispositivi pieghevoli

I dispositivi pieghevoli stanno prendendo sempre più piede nel mercato degli smartphone, gli sviluppatori ed i designer devono essere pronti a rivoluzionare le loro applicazioni per sfruttare al meglio le nuove funzionalità portate da questi dispositivi.

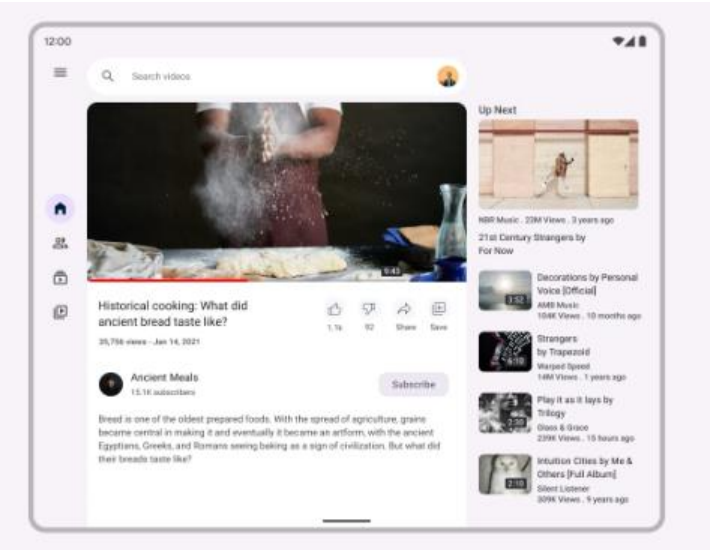
Material Design 3 punta a rendere più facile la transizione verso i dispositivi pieghevoli con un nuovo insieme di linee guida, composizioni ed animazioni.

Composizioni

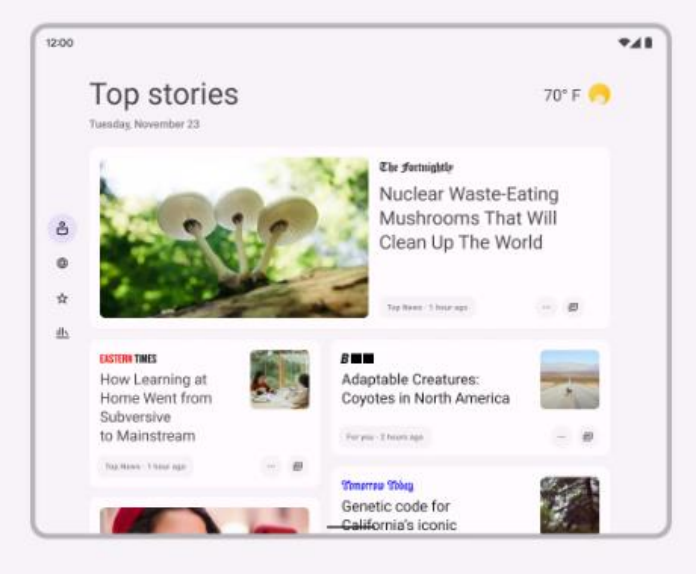
Material 3 mette a disposizione un insieme di nuovi layout dedicati agli schermi grandi e pieghevoli.



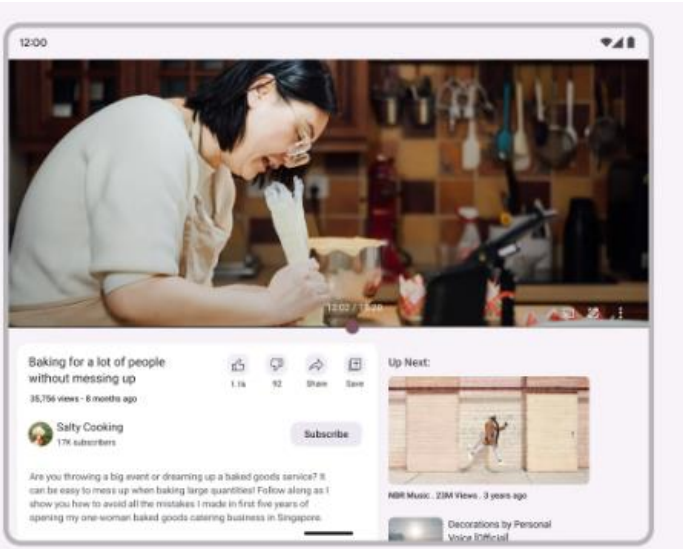
Lists, per mostrare informazioni sulla sinistra dello schermo ed espanderne i dettagli nella parte destra



Supporting Panels, molto simile a Lists ma il pannello principale stavolta è nella parte destra



Feeds, per mostrare nuovi contenuti



Hero, per risaltare le immagini o video nella parte superiore dello schermo

Animazioni

I dispositivi pieghevoli consentono interazioni e transizioni completamente nuove. Ad esempio, ora più app possono essere eseguite contemporaneamente e gli utenti possono passare costantemente da dimensioni dello schermo radicalmente diverse. Per esempio sono disponibili nuove animazioni per:

navigation bars, apertura schermo in landscape e apertura di dialogs.

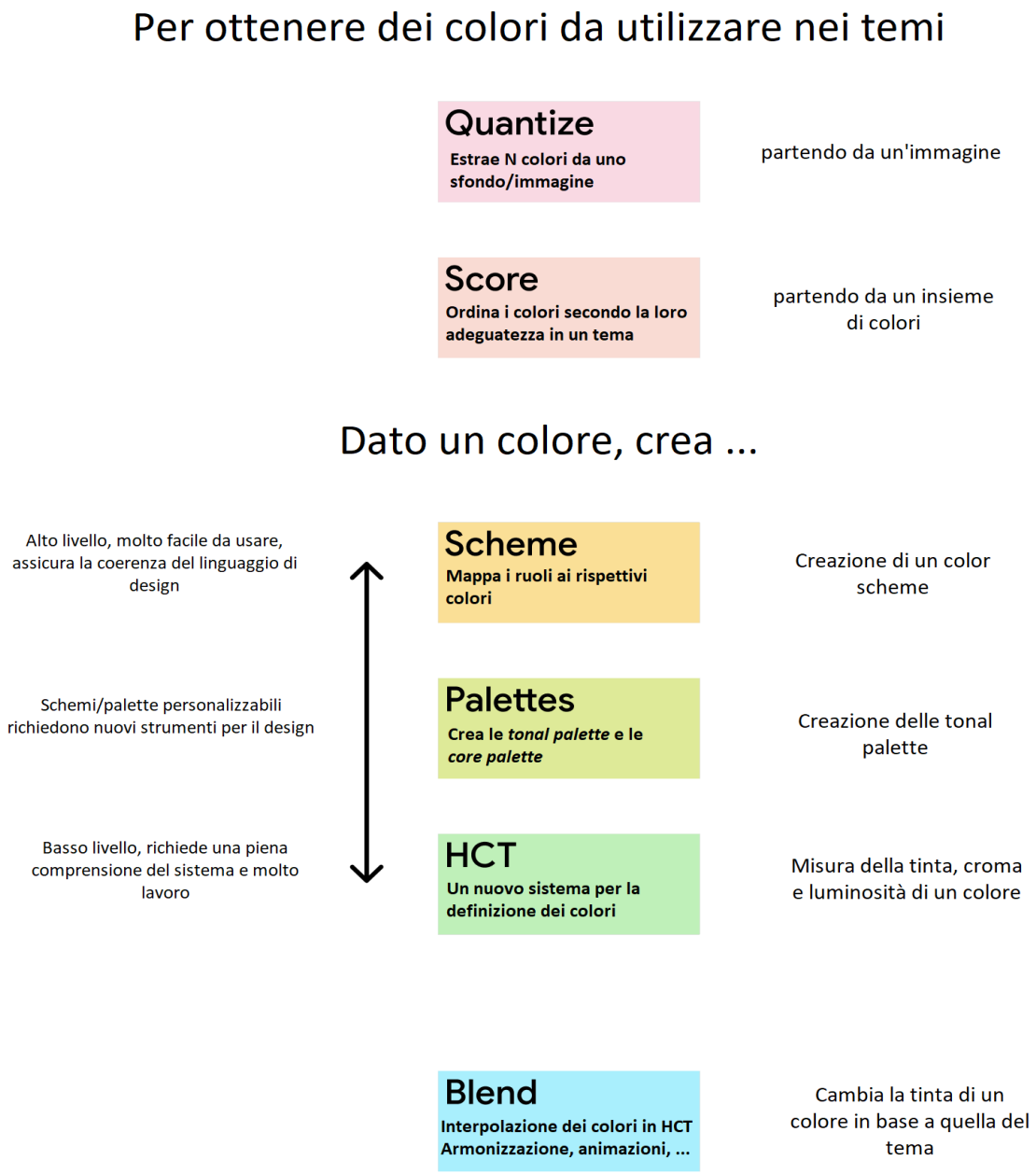
DYNAMIC COLORS ALGORITHMS

Material Foundation, l'organizzazione che sta dietro a Material Design, mette a disposizione su GitHub una repository chiamata "material-color-utilities".

Material Color Utilities è una libreria multipiattaforma che contiene tutto il necessario per implementare Material You. Fu inizialmente creata per i Pixel per poi passare alle altre case di produzione di dispositivi Android.

La libreria è open source, chiunque può contribuire a migliorarne gli algoritmi. Il progetto è in continuo aggiornamento, in un futuro prossimo la libreria verrà aggiornata per incrementare il numero di moduli e le piattaforme supportate (iOS ad esempio). Al momento le librerie disponibili sono in Dart, Java e Typescript.

Material Color Utilities comprende queste 6 componenti:



Andremo ora ad analizzare i componenti **Quantize**, **Scheme** e **Palette**, ritenuti i più interessanti ai fini del report.

Prima di iniziare però è bene rendere noto che Material You introduce un nuovo sistema per la rappresentazione dei colori chiamato HCT. Questo sistema si basa sull'individuazione di colori mediante

- **Hue,** ovvero la tonalità (lunghezza d'onda della radiazione luminosa)
- **Chroma**, ovvero la croma, il grado di intensità del colore
- **Tone,** che indica la luminosità del colore.

Nel codice dei componenti vedremo che i colori saranno espressi tramite un numero intero, grazie alla trasformazione della codifica ARGB (in esadecimale).

Quantize

Questa funzione permette di estrapolare N colori da un immagine, quest'ultimi verranno poi utilizzati per la creazione delle palette.

Questo pacchetto comprende diverse classi, ognuna con una specifica funzione.

La prima che andiamo ad analizzare è QuantizerCelebi, un quantizzatore di immagini che migliora la qualità di un normale algoritmo K-mean mettendo come stato iniziale il risultato di un altro quantizzatore(Wu quantizer al posto di centroidi (punti medi) casuali.

Inoltre grazie a molte ottimizzazioni, implementate nella classe Wsmeans, ha prestazioni ottime.

Questo algoritmo è stato creato da M. Emre Celebi e ripreso dagli sviluppatori di Material Color Utils dal documento "Improving the Performance of K-Means for Color Quantization" (2011).

Algoritmo K-Means:

Prima di analizzare il codice sorgente di questa classe è meglio dare una definizione, in maniera generale, di algoritmo K-means.

L'algoritmo **K-*means*** è un algoritmo di analisi dei gruppi partizionale che permette di suddividere un insieme di oggetti in *k* gruppi sulla base dei loro attributi. È una variante dell'algoritmo di aspettativa-massimizzazione (EM) il cui obiettivo è determinare i *k* gruppi di dati generati da distribuzioni gaussiane. Si assume che gli attributi degli oggetti possano essere rappresentati come vettori, e che quindi formino uno spazio vettoriale. L'obiettivo che l'algoritmo si prepone è di minimizzare la varianza totale intra-gruppo.

Ogni gruppo viene identificato mediante un *centroide* o *punto medio*. L'algoritmo segue una procedura iterativa:

1) Inizialmente crea *k* partizioni e assegna i punti d'ingresso a ogni partizione o casualmente o usando alcune informazioni euristiche. Nel nostro algoritmo di Celebi inizierà dal risultato di un altro algoritmo di quantizzazione (WuQuantizer).
2) Quindi calcola il centroide di ogni gruppo.
3) Costruisce in seguito una nuova partizione associando ogni punto d'ingresso al gruppo il cui centroide è più vicino ad esso.
4) Infine vengono ricalcolati i centroidi per i nuovi gruppi e così via, finché l'algoritmo non converge.

Possiamo ora riportare il codice sorgente:

QuantizerCelebi.java

```
package quantize;

import java.util.Map;
import java.util.Set;
public final class QuantizerCelebi {
 private QuantizerCelebi() {}


 public static Map<Integer, Integer> quantize(int[] pixels, int maxColors) {
  QuantizerWu wu = new QuantizerWu();
  QuantizerResult wuResult = wu.quantize(pixels, maxColors);

  Set<Integer> wuClustersAsObjects = wuResult.colorToCount.keySet();
  int index = 0;
  int[] wuClusters = new int[wuClustersAsObjects.size()];
  for (Integer argb : wuClustersAsObjects) {
   wuClusters[index++] = argb;
  }

  return QuantizerWsmeans.quantize(pixels, wuClusters, maxColors);
 }
}
```

Come si può notare dal codice la struttura della classe è molto semplice. Essa presenta un solo metodo:

```
public static Map<Integer, Integer> quantize(int[] pixels, int maxColors
```

che si occupa di ridurre il numero di colori necessari per rappresentare l'input, limitando al minimo la differenza tra

l'immagine originale e l'immagine ricolorata.

I parametri in ingresso del metodo sono:

- **pixels**, un array di numeri interi che contiene il colore, codificato con un numero intero, di tutti i pixel dell'immagine di partenza.
- **maxColors**, un numero intero che indica il numero di colori in cui si vuole dividere un'immagine.

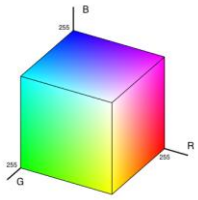
restituisce un oggetto di tipo **Map<Integer,Integer>** nel quale le chiavi saranno colori espressi nel formato ARGB, mentre il valore ad esse associato sarà il numero di pixel dell'immagine originale che corrispondono al colore della chiave, presente nell'immagine quantizzata.

In primo luogo si crea un oggetto di tipo QuantizerResult che rappresenta il risultato di un'operazione di quantizzazione ovvero una Mappa di valori di tipo e chiavi di tipo Integer, esso è definito nella seguente classe:

QuantizerResult.java

```
package quantize;

import java.util.Map;

public final class QuantizerResult {
  public final Map<Integer, Integer> colorToCount;

  QuantizerResult(Map<Integer, Integer> colorToCount) {
    this.colorToCount = colorToCount;
  }
}
```

L'oggetto di tipo QuantizerResult viene inizializzato con il risultato ottenuto usando il metodo **quantize** di un altro quantizzatore: QuantizerWu.



Il codice della classe QuantizerWu è un quantizzatore che suddivide i pixel di un'immagine in clusters dividendo progressivamente un "cubo RGB", questi cluster si basano sul peso dei pixel in ogni area del cubo.

Non ci soffermeremo a descrivere ulteriormente questa classe data l'elevata complessità e lunghezza del suo codice.

L'aspetto importante del QuantizerWu è che fornisce uno stato (un array di colori chiamato **wuClusters**) dal quale l'algoritmo K-mean vero e proprio presente nella classe QuantizerWsmeans inizia ad operare, garantendo alte prestazioni.

La presenza di questo stato iniziale non è necessaria, QuantizerWsmeans funzionerebbe comunque ma si verificherebbe un calo della sua velocità.

Tornando a **quantize** di **QuantizerCelebi.java** vediamo che come ultima azione crea un oggetto di tipo QuantizerWsmeans e chiama il suo metodo quantize. Il risultato di questo metodo sarà poi il risultato finale della classe.

Per quanto riguarda la classe QuantizerWsmeans, anch'essa risulta molto complicata a livello logico e ci limiteremo a descrivere la sua funzione generale.

QuantizerWsmeans è un quantizzatore di immagini che migliora le prestazioni di un algoritmo K-mean standard, grazie a svariate ottimizzazioni come la deduplicazione di pixel identici oppure l'utilizzo della disuguaglianza triangolare per ridurre il numero di confronti che un punto deve effettuare.

Anche questo algoritmo, chiamato Weighted Square Means, è stato ideato da M. Emre Celebi ed è stato ripreso dalla sua pubblicazione "Improving the Performance of K-Means for Color Quantization"(2011).Il metodo principale di questa classe è ovviamente **quantize**,

```
public static Map<Integer, Integer> quantize(
int[] inputPixels, int[] startingClusters, int maxColors)
```

I parametri in ingresso sono gli stessi della classe **QuantizerCelebi** con l'aggiunta del parametro **startingClusters,** il quale è un array di colori codificati che rappresenta il risultato di **QuantizerWu.**

Questo metodo restituirà una mappa con chiavi un insieme di colori in formato ARGB, mentre i valore ad esse associato sarà il numero di pixel di input dello stesso colore della chiave.

Questa mappa verrà poi restituita dal metodo QuantizerCelebi per poi divenire il risultato ultimo di tutte le operazioni del pacchetto **Quantize.**

Questi metodi di quantizzazione permettono quindi la creazione delle palette di colori dinamici. I colori presenti nell'insieme delle chiavi della mappa restituita da **QuantizerCelebi** saranno infatti l base per un altro componente di Material Utils: **Score.**

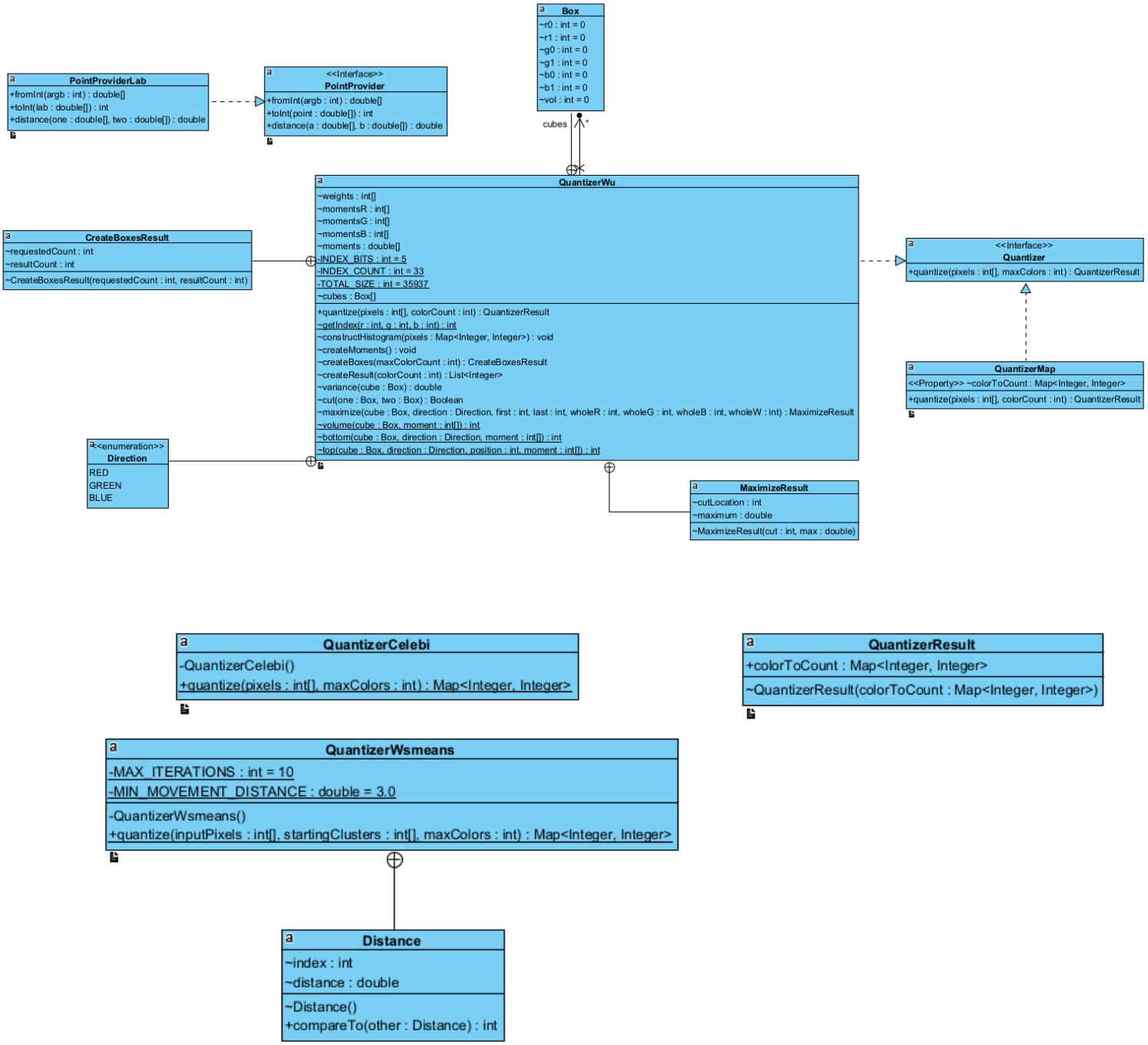
Questo componente si occuperà di filtrare i vari colori di questa mappa e ordinarli in una classifica, dal più adatto al theming a quello meno opportuno.

Come abbiamo visto in precedenza questo è il fondamento per la funzionalità dei colori dinamici.

Quantize e **Score** sono già integrati all'interno della libreria Material Design Components (MDC).

Diagramma UML

Riportiamo ora il diagramma UML del pacchetto, dove vengono mostrate altre classi ed interfacce di cui abbiamo omesso la descrizione.



Scheme

Questo componente di Material Color Utils è molto più semplice del precedente, si tratta infatti di una classe chiamata **Scheme** che memorizza in sé stessa tutti i mapping ruolo-colore scelti per il sistema di design.

Nella classe ci sarà infatti un attributo privato per ogni color role che verrà inizializzato al momento della creazione dell'oggetto.

Scheme.java

```
public class Scheme {
  private int primary;
  private int onPrimary;
  private int primaryContainer;
  private int onPrimaryContainer;
  private int secondary;
  private int onSecondary;
  private int secondaryContainer;
  private int onSecondaryContainer;
  private int tertiary;
  private int onTertiary;
  private int tertiaryContainer;
  private int onTertiaryContainer;
  private int error;
  private int onError;
  private int errorContainer;
  private int onErrorContainer;
  private int background;
  private int onBackground;
  private int surface;
  private int onSurface;
  private int surfaceVariant;
  private int onSurfaceVariant;
  private int outline;
  private int shadow;
  private int inverseSurface;
  private int inverseOnSurface;
  private int inversePrimary;

...
```

Il costruttore di un oggetto Scheme infatti avrà 27 parametri di ingresso, uno per ogni ruolo, di tipo Int; il numero inserito sarà la codifica in ARGB del colore scelto per il ruolo corrispondente.

Viene riportato il codice del costruttore della classe:

```
public Scheme(
    int primary,
    int onPrimary,
    int primaryContainer,
    int onPrimaryContainer,
    int secondary,
    int onSecondary,

            ...                                          //omesse per ragioni di spazio

    int inverseOnSurface,
    int inversePrimary) {
  super();
  this.primary = primary;
  this.onPrimary = onPrimary;
  this.primaryContainer = primaryContainer;
  this.onPrimaryContainer = onPrimaryContainer;
  this.secondary = secondary;
  this.onSecondary = onSecondary;

              ...

  this.inverseOnSurface = inverseOnSurface;
  this.inversePrimary = inversePrimary;
}
```

Per ogni color role inoltre vengono forniti i metodi getter e setter.

Infine questa classe presenta dei metodi molto interessanti per creare color schemes partendo da un colore codificato in argb. I metodi in questione sono i seguenti:

```
public static Scheme light(int argb) {
  return lightFromCorePalette(CorePalette.of(argb));
}

public static Scheme dark(int argb) {
  return darkFromCorePalette(CorePalette.of(argb));
}

public static Scheme lightContent(int argb) {
  return lightFromCorePalette(CorePalette.contentOf(argb));
}

public static Scheme darkContent(int argb) {
  return darkFromCorePalette(CorePalette.contentOf(argb));
}
```

Come si può vedere dal codice questi metodi chiamano a loro volta dei metodi privati, che prendendo come parametro un oggetto di tipo “CorePalette” (descritto in seguito), restituiscono un’oggetto di tipo Scheme con tutti i ruoli già mappati.

Esaminiamo ora solo il metodo **lightFromCorePalette** dato che gli altri tre si basano sullo stesso principio

```
private static Scheme lightFromCorePalette(CorePalette core) {
  return new Scheme()
      .withPrimary(core.a1.tone(40))
      .withOnPrimary(core.a1.tone(100))
      .withPrimaryContainer(core.a1.tone(90))
      .withOnPrimaryContainer(core.a1.tone(10))
      .withSecondary(core.a2.tone(40))
      .withOnSecondary(core.a2.tone(100))
      .withSecondaryContainer(core.a2.tone(90))
      .withOnSecondaryContainer(core.a2.tone(10))
      .withTertiary(core.a3.tone(40))
      .withOnTertiary(core.a3.tone(100))
      .withTertiaryContainer(core.a3.tone(90))
      .withOnTertiaryContainer(core.a3.tone(10))
      .withError(core.error.tone(40))
      .withOnError(core.error.tone(100))
      .withErrorContainer(core.error.tone(90))
      .withOnErrorContainer(core.error.tone(10))
      .withBackground(core.n1.tone(99))
      .withOnBackground(core.n1.tone(10))
      .withSurface(core.n1.tone(99))
      .withOnSurface(core.n1.tone(10))
      .withSurfaceVariant(core.n2.tone(90))
      .withOnSurfaceVariant(core.n2.tone(30))
      .withOutline(core.n2.tone(50))
      .withShadow(core.n1.tone(0))
      .withInverseSurface(core.n1.tone(20))
      .withInverseOnSurface(core.n1.tone(95))
      .withInversePrimary(core.a1.tone(80));
}
```

Questo metodo accetta in ingresso un oggetto di tipo CorePalette che, come vedremo in seguito, conterrà i cinque **key color** che origineranno il color scheme.

Nel metodo **lightFromCorePalette** a questi colori chiave verrà poi variata la luminosità e assegnata ad un ruolo specifico.

Ad esempio la riga

```
.withPrimaryContainer(core.a1.tone(90))
```

Assegnerà al ruolo di PrimaryContainer il colore **a1** (accent primary) con grado di luminosità pari a 90.

L’oggetto completo verrà restituito al metodo **light** che a sua volta lo ritornerà al chiamante.

Diagramma UML

Palette

L'ultimo componente che andremo ad analizzare sarà **Palettes,** quest'ultimo è composto da due classi:

CorePalette e **TonalPalette**.

TonalPalette

Si tratta di una classe di supporto che si occupa di fornire colori che sono costanti in crominanza e tinta ma presentano valori di luminosità differenti. Ecco il codice della classe **TonalPalette.java**:

```
public final class TonalPalette {
  Map<Integer, Integer> cache;
  double hue;
  double chroma;
```

Come vediamo nella classe ci sono due attributi atti a salvare la crominanza (chroma) e la tinta (hue) che caratterizzeranno ogni colore appartenente alla Tonal Palette. I metodi della classe riportati in seguito serviranno a creare un oggetto di TonalPalette partendo da un singolo colore, da cui verranno estratti chroma e hue, oppure per inserimento diretto dei due parametri.

```
  public static final TonalPalette fromInt(int argb) {
    Hct hct = Hct.fromInt(argb);
    return TonalPalette.fromHueAndChroma(hct.getHue(), hct.getChroma());
  }

  public static final TonalPalette fromHueAndChroma(double hue, double chroma) {
    return new TonalPalette(hue, chroma);
  }

  private TonalPalette(double hue, double chroma) {
    cache = new HashMap<>();
    this.hue = hue;
    this.chroma = chroma;
  }
```

Il metodo più interessante della classe è **tone**, il quale permette di inserire il valore di luminosità desiderato (intero da 0 a 100) come parametro e da questo ottenere il colore della palette corrispondente.

Nella classe TonalPalette non saranno salvati direttamente tutti i colori per ogni grado di luminosità ma verranno creati direttamente, grazie al sistema HCT, e aggiunti di volta in volta ad una mappa che funziona da cache.

Questa cache porterà un notevole aumento delle prestazioni.

```
@SuppressWarnings("ComputeIfAbsentUseValue")
public int tone(int tone) {
  Integer color = cache.get(tone);
  if (color == null) {
    color = Hct.from(this.hue, this.chroma, tone).toInt();
    cache.put(tone, color);
  }
  return color;
}
```

CorePalette

Un oggetto di tipo CorePalette invece è un insieme di TonalPalettes necessarie per creare i color schemes.

Come possiamo vedere dal codice la classe presenterà sei attributi pubblici, si tratta di sei diverse TonalPalette associate ad un singolo colore. Cinque di questi colori (**a1,a2,a3,n1,n2**) saranno esattamente i **key colors** che daranno vita al color scheme, il restante attributo servirà per creare la palette per i messaggi di errore.

```
public final class CorePalette {
  public TonalPalette a1;
  public TonalPalette a2;
  public TonalPalette a3;
  public TonalPalette n1;
  public TonalPalette n2;
  public TonalPalette error;
```

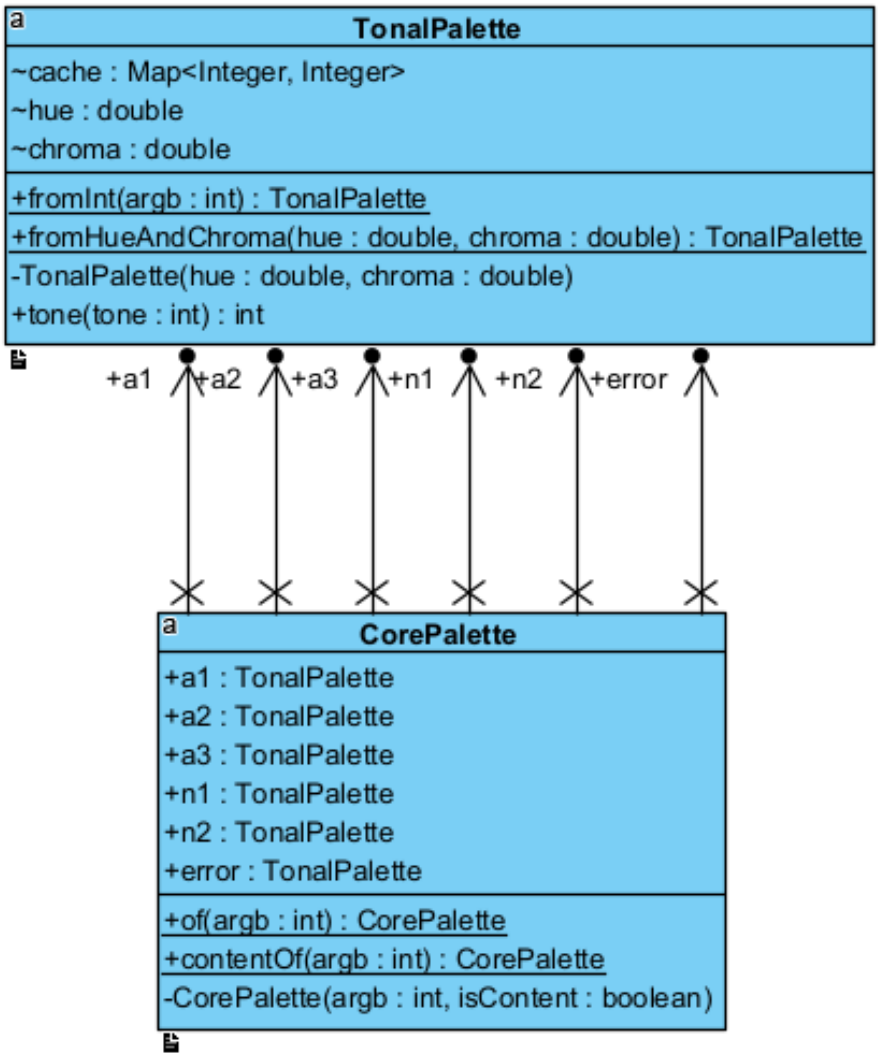
I metodi **of** e **contentOf** restituiscono, partendo da un colore codificato in ARGB, un oggetto di tipo CorePalette con al suo interno le TonalPalette create per ognuno dei key color (+ il colore di errore).

Queste palette verranno generate grazie al metodo **fromHueAndChroma** della classe TonalPalette, di cui il codice è stato riportato in precedenza.

```
  public static CorePalette of(int argb) {
    return new CorePalette(argb, false);
  }

  public static CorePalette contentOf(int argb) {
    return new CorePalette(argb, true);
  }
```

```
  private CorePalette(int argb, boolean isContent) {
    Hct hct = Hct.fromInt(argb);
    double hue = hct.getHue();
    double chroma = hct.getChroma();
    if (isContent) {
      this.a1 = TonalPalette.fromHueAndChroma(hue, chroma);
      this.a2 = TonalPalette.fromHueAndChroma(hue, chroma / 3.);
      this.a3 = TonalPalette.fromHueAndChroma(hue + 60., chroma / 2.);
      this.n1 = TonalPalette.fromHueAndChroma(hue, min(chroma / 12., 4.));
      this.n2 = TonalPalette.fromHueAndChroma(hue, min(chroma / 6., 8.));
    } else {
      this.a1 = TonalPalette.fromHueAndChroma(hue, max(48., chroma));
      this.a2 = TonalPalette.fromHueAndChroma(hue, 16.);
      this.a3 = TonalPalette.fromHueAndChroma(hue + 60., 24.);
      this.n1 = TonalPalette.fromHueAndChroma(hue, 4.);
      this.n2 = TonalPalette.fromHueAndChroma(hue, 8.);
    }
    this.error = TonalPalette.fromHueAndChroma(25, 84.);
  }
}
```

Diagramma UML



La nostra applicazione: DiarioM3

Per mostrare le caratteristiche principali di Material Design 3 è stata importante la scelta dell'applicazione.
Abbiamo cercato di esprimere i 3 principi cardine del Material You(Comfortable,Iconoclastic,Spirited) in un'applicazione per l'autogestione personale. ***DiarioM3*** è infatti un "diario alimentare", che aiuta l'utente a memorizzare le calorie assunte nel corso della giornata.

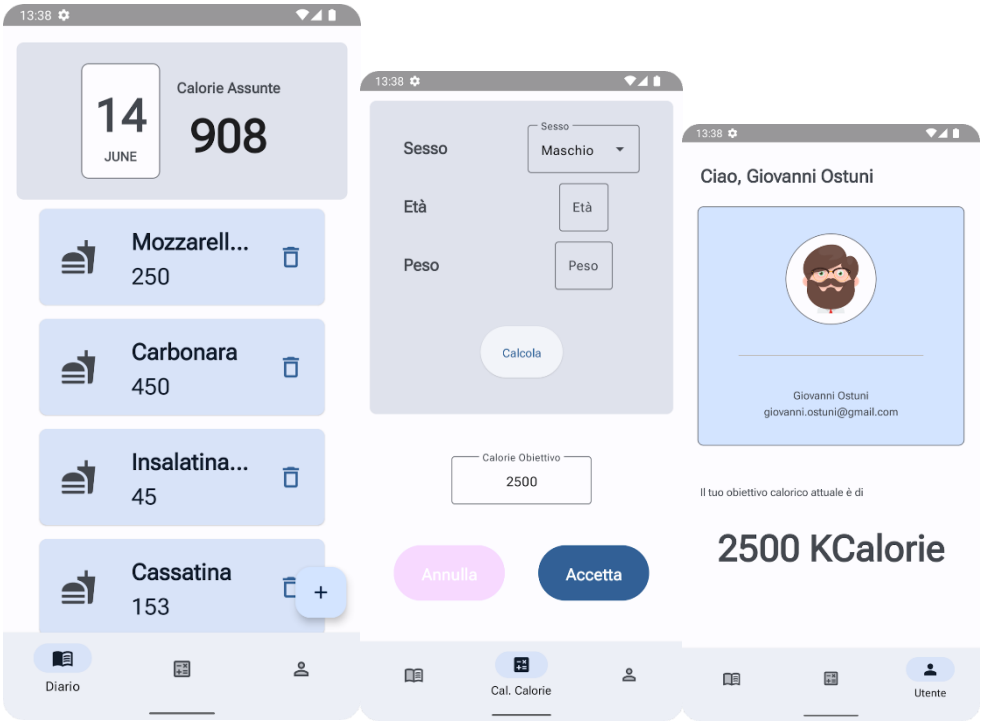
La scelta di un "diario alimentare" ha reso più immediata e intuitiva la comunicazione del primo principio: comfortable.
Il termine *comfortable* è stato introdotto da Christian Robertson nel video di presentazione come: *How can a design system can make a person feel home with his device*.
Per rappresentare questo abbiamo voluto che la nostra applicazione potesse essere intuitiva e semplice, unendo il design ad un'esperienza di utilizzo facile ed immediata. Questo ha lo scopo di rendere ***DiarioM3*** uno strumento che possa essere usato quotidianamente.
Abbiamo, infatti,sviluppato un'interfaccia priva di menu incapsulati e nascosti. Come vedremo tutte le informazioni e i pulsanti sono ben visibili e intuitivi.

L'aspetto Iconoclastic, definito come, *How can software exhibit the awareness to adapt and thrive through changes, blurring boundaries between sys apps and installed apps,*è stato invece trasmesso grazie alla combinazione di colori dinamici ed prendendo ispirazione da applicazioni di sistema

Il principio "Spirited" è, infine, *imbue digital development with the spirit of the natural world, sense of aliveness through shapes and motions*. Questo è stato raggiunto grazie ad elementi di layout come le *Material Cards* oppure i diversi stili di bottoni, che rendono le superfici più naturali, emulando materiali del mondo reale come ad esempio la schermata principale, che cerca di rappresentare un calendario stampato senza elementi in rilievo.

Struttura

DiarioM3 è composta da 3 da tre pagine:*diario, cal. calorie* e *profilo*.
Nella pagina **“Diario”**, la pagina principale, il layout rappresenta un calendario con la data e la somma delle calorie assunte in quella giornata.
Al di sotto si trova la lista degli alimenti mangiati, con le relative calorie associate. Da questa schermata si possono aggiungere alimenti alle liste selezionando il pulsante in rilievo contrassegnato con l'icona “+”. Scorrendo a destra o a sinistra si possono visualizzare i giorni futuri e passati con i relativi alimenti.
La seconda schermata, “***Calc. Calorie***” è utile per segnare l'obiettivo calorico giornaliero. È inoltre presente una calcolatrice del fabbisogno calorico che utilizza un’equazione indicata dall'OMS per ottenere il *Metabolic Equivalent of Task* a riposo.
L'ultima pagina, “Profilo”, racchiude i dati principali dell'utente che sono: l'obiettivo calorico, il nome e la mail.



DiarioM3 schermate principali

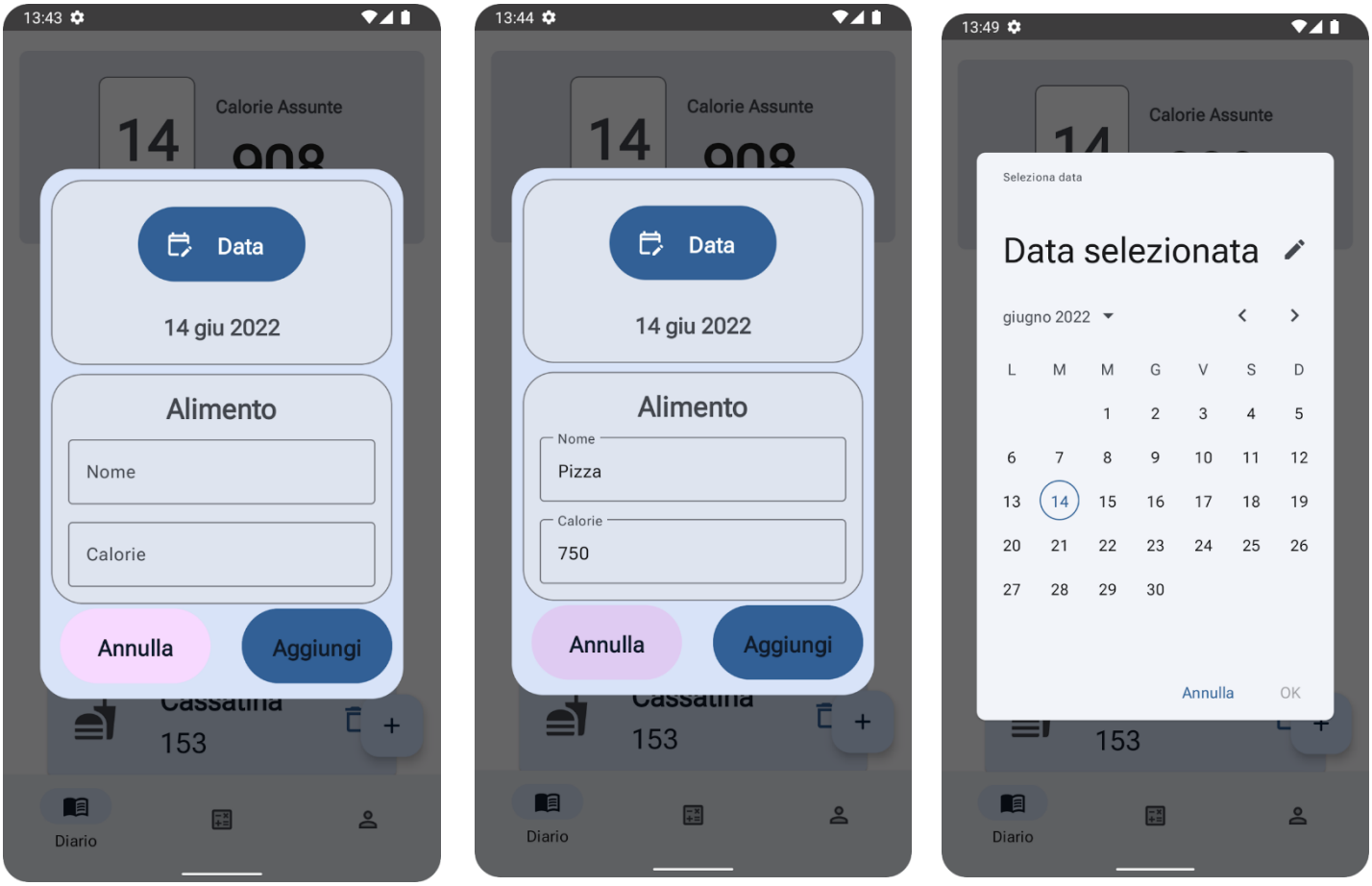
Funzionalità e Design

In questa sezione andremo ad esporre le principali funzionalità della nostra applicazione

1. **Aggiunta degli alimenti**

   Nella schermata "*Diario*" è presente un tasto in rilievo che permette di aggiungere un alimento.
   Questo, di default, viene aggiunto il giorno stesso,ma grazie al tasto "*Data*" c'è la possibilità di aggiornare questo valore con quello desiderato.
   Tramite il tasto aggiungi si conclude l'inserimento.
   Questo *DialogFragment* ha presenti nel suo layout molti componenti descritti dal material design 3:

   - *MaterialCardView* in particolare di tipo *Outlined card*
   - Tre *Button* di tipo *Filled button*
   - Due input di testo di tipo *Outlined text field*
   - Un *MaterialDatePicker*



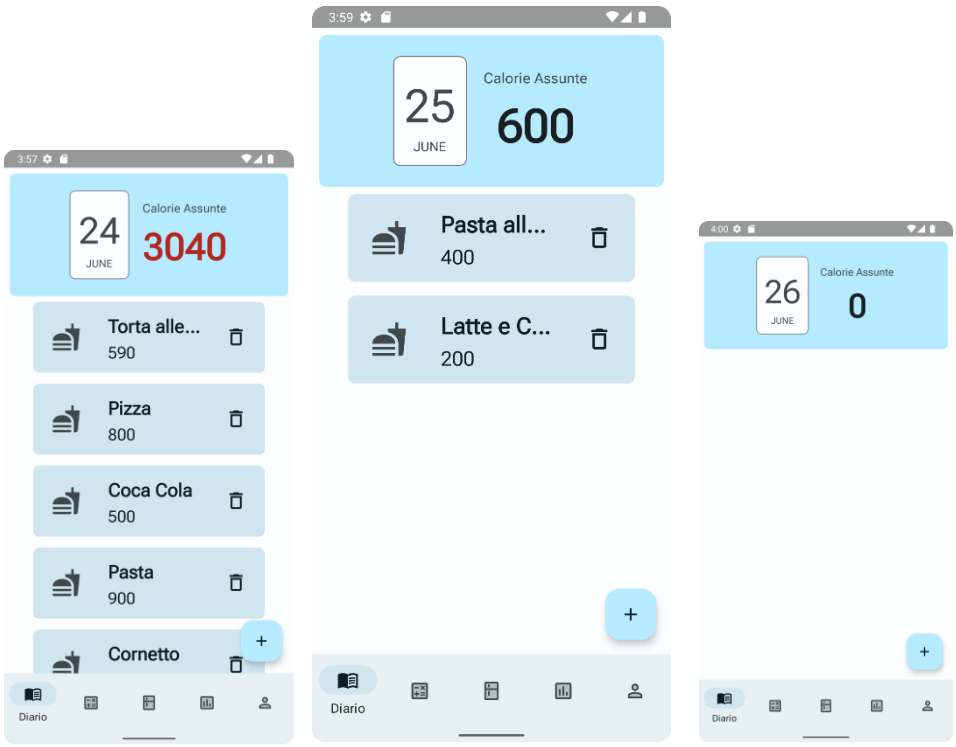
DiarioM3 schermata di aggiunta alimento

2.Eliminazione degli alimenti e panoramica della schermata Diario

La schermata principale è formata da un *ViewPager* che permette di scorrere all'infinito a destra ed a sinistra per visualizzare i giorni passati e futuri, come un calendario classico.
Grazie a questa pagina abbiamo la possibilità di visualizzare la lista e le calorie assunte durante la giornata.
Questo contatore ha inoltre la possibilità di segnalare quando l'obiettivo calorico, impostato dall'utente, viene superato segnando le calorie assunte con un colore "*Error*".
Gli alimenti hanno la possibilità di essere eliminati premendo il tasto "cestino" riposto a destra di ogni tessera.

I componenti di material design 3 presenti in questo *Fragment* sono:

- *MaterialCardView* in particolare di tutti e tre i tipi:*Filled card, Outlined card* ed *Elevated Card*
- Un *FloatingActionButton*
- Un *Icon text button* per ogni alimento



DiarioM3 schermata Diario

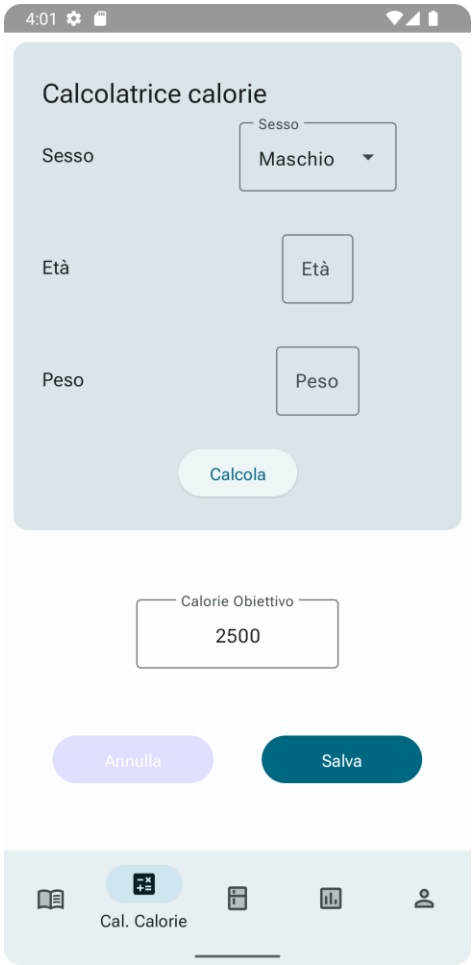
3.Schermata Cal. Calorie

La schermata *Cal. Calorie* è formata da una calcolatrice, per stimare il fabbisogno energetico.Questa utilizza un'equazione fornita dall'*OMS* negli anni '80.
La seconda metà di questa schermata invece dà la possibilità di modificare ed aggiornare il proprio obiettivo calorico. Questo dato persistente viene aggiornato e condiviso con tutte le pagine dell'applicazione.
I componenti di material design 3 presenti in questo *Fragment* sono:

- *MaterialCardView* in di tipo *Filled card*
- Un *Button* di tipo *Elevated*
- Due *Button* di tipo *Filled*
- Un *Menu* per selezionare il genere
- Tre input di test di tipo *Outlined*

Un ulteriore elemento di material design 3 chiaramente presente all'interno di questa pagina è l'utilizzo preciso dei colori.
Grazie alle palette create dinamicamente da Android si possono differenziare i bottoni e le sezioni dei layout in maniera chiara all'utente finale.
Un chiaro caso di differenziazione di colori è presente nei 3 bottoni.

Il bottone "*Calcola*" ha lo stesso colore base dello sfondo, così da comunicare all'utente la sua minor importanza rispetto ai due bottoni sottostanti.



Successivamente il bottone "*Annulla*", descritto con il colore "*colorTertiaryContainer*" suggerisce un'importanza di secondo livello, ma comunque rilevante per l'utilizzo di questa schermata

Infine il più importante, il Bottone "*Accetta*", spicca sui 3 grazie all'utilizzo del colore primario della palette.

DiarioM3 schermata Cal. Calorie

4.Schermata Cal. Calorie

La schermata *Grafici* è utile all'utente per visualizzare e condividere il resoconto dei suoi inserimenti nell'arco temporale di un mese.
I dati sono stati rappresentati su 3 diverse grafiche che utilizzano i colori dinamici.
Il primo è un grafico a ciambella che è integrata dai progressBar, utilizzati come legenda.
L'ultimo invece è un istogramma che rappresenta giornalmente gli offset dell'obiettivo. In questo istogramma abbiamo aggiunto pochi dati ma immediati.
Sono infatti presenti al massimo due numeri: le differenze di calorie nei giorni in cui ti sei allontanato di più per eccesso e per difetto dell'obiettivo.
Tutta questa schermata infine ha la possibilità di essere condivisa sulle piattaforme preferite, con un messaggio personalizzato.
I componenti di material design 3 presenti in questo *Fragment* sono:

- *MaterialCardView* in di tipo *Elevated card*
- Un dialog per l'inserimento della data

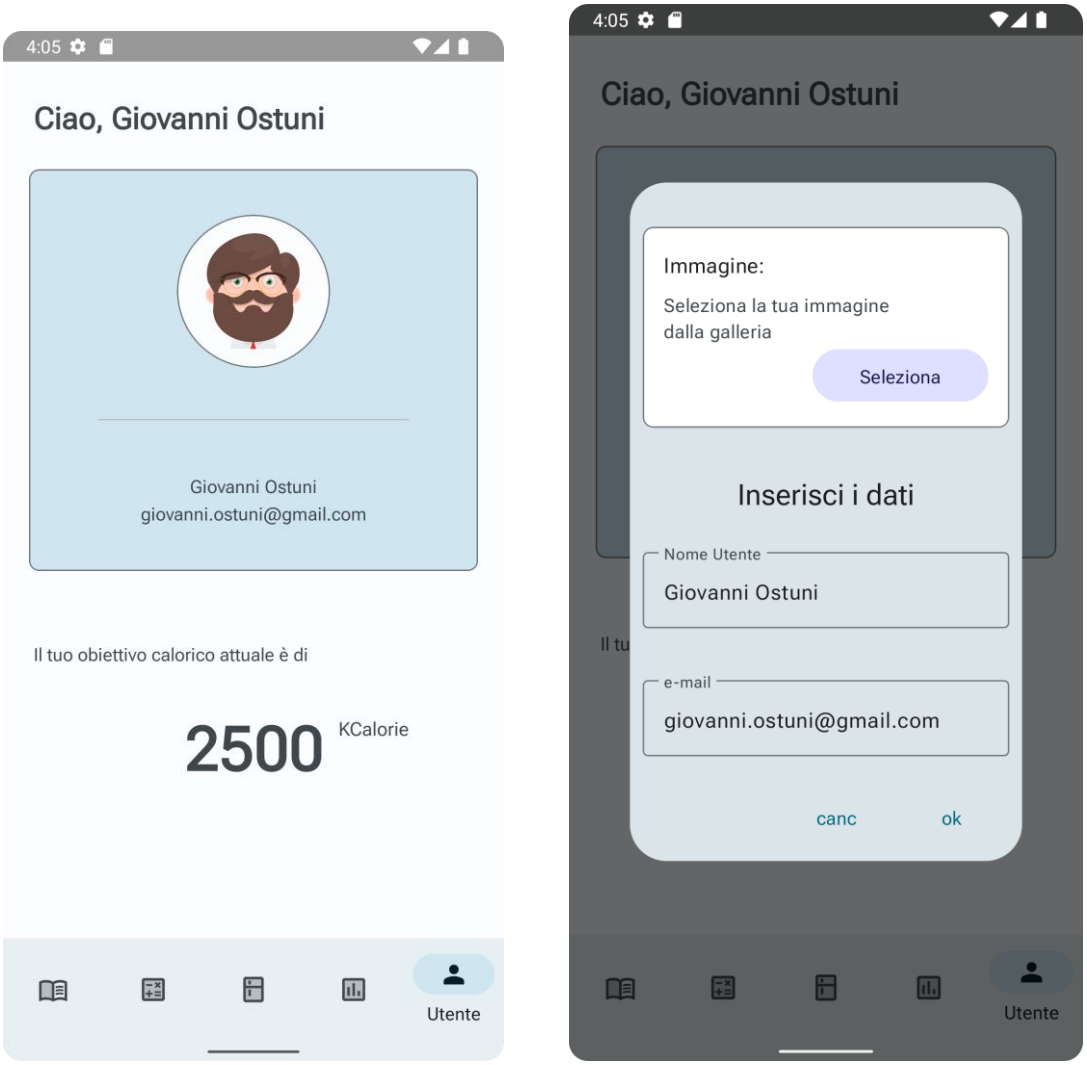
DiarioM3 schermata Grafici

5.Schermata Utente

La schermata *Utente* è l'ultima pagina di questa applicazione .Ha il compito di mostrare all'utente le informazioni principali inserite dallo stesso.
Al suo interno è possibile modificare il nome e la mail accedendo ad un *DialogFragment* che appare cliccando sulle stesse.
I componenti di Material design 3 presenti in questa sezione sono:

- *MaterialCardView* in di tipo *Outlined*
- Due Button di tipo *Text Button*
- Due input di test di tipo *Outlined*

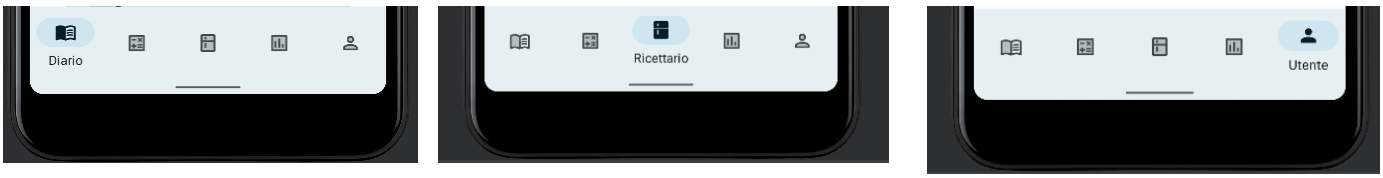
Ancora una volta, l'utilizzo di elementi di design come i *Text Button* rendono più intuitiva e semplice l'applicazione. Questi pulsanti, infatti, suggeriscono all'utente che le informazioni immesse non sono rilevanti per il corretto utilizzo dell'applicazione a differenza dei dialog e pulsanti precedentemente mostrati.



DiarioM3 schermata Utente

6.Navigation bar

Per muoversi tre le schermate ***DiarioM3*** utilizza una Navigation Bar.
Questo elemento ha ricevuto particolari attenzione nel passaggio da MaterialDesign 2 al 3.
Come spiegato nella documentazione del Material Design 3, questa deve seguire particolari direttive.
Abbiamo quindi implementato quelle che secondo noi sono le principali.
Questa barra accessibile in qualunque momento racchiude in modo efficace le informazioni di ogni schermata.
Quando viene selezionata la schermata un'animazione ,che si svolge in un solo asse, in questo caso nell'asse orizzontale, crea un'asola colorata intorno all'icona.
Quest'ultima passa da essere *Twotone* ad una *Filled* per rendere la selezione ancora più evidente.
Infine viene mostrato il titolo della schermata selezionata che fornisce una breve descrizione della pagina.
Volendo concentrarci al massimo sulla pulizia del design, abbiamo deciso di rendere il tasto gesture, sul fondo dello schermo, integrato al nostro design. Il motivo principale di questa scelta è dovuto allo spirito *Iconoclastic* dettato da *Christian Robertson*.
Per essere coerenti al massimo con questa filosofia abbiamo reso lo status bar *translucid*.



DiarioM3 NavigationBar e gestureButton

DiarioM3 StatusBar

7.Ricettario

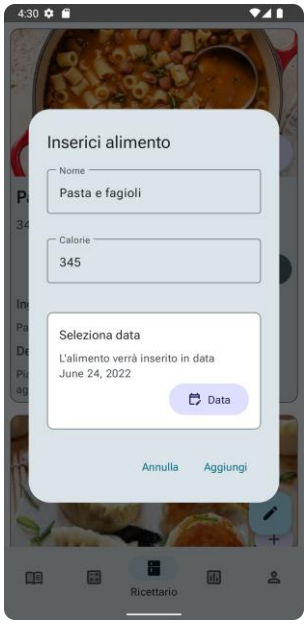
Questa schermata ha la funzionalità di raccogliere determinate ricette, inserite dall'utente, in modo da poterle inserire direttamente nella scermata *Diario*.

La schermata prevede una Recycler View i cui gli elementi sono delle Cards implementate seguendo le guidelines di Material Design.

Ognuna di queste Card può essere toccate per rivelare maggiori dettagli sulla ricetta. Abbiamo inserito una semplice animazione nella comparsa di questi dettagli.

L'utente inoltre può aggiungere delle immagini personalizzate a queste Cards che si aggiorneranno in automatico.

Come accennato in precedenza, il pulsante "+" presente in ogni *Card* permette di aggiungere velocemente la ricetta alla schermata Diario, alla premuta di questo bottone si aprirà infatti un dialog che permetterà di aggiungere l'alimento in un qualsiasi giorno.



Elementi principali di Material Design 3 e Implementazione

Come elencato prima, gli elementi del material che sono presenti all'interno dei layout della nostra applicazione sono principalmente:

1. Navigation bar
2. Button
3. Cards
4. Text fields

Le implementazioni di questi componenti sono avvenute in questo modo:

1. Navigation bar: questa è stata implementata all'interno della file *activity_main.xml* nel seguente modo

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottom_navigation_main"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:menu="@menu/activity_main_drawer"
    app:labelVisibilityMode="selected"
    app:itemRippleColor="@android:color/transparent"/>
```

in particolare la line app:labelVisibilityMode="selected" serve per far si che le label vengano nascoste quando non è selezionata la schermata.
app:itemRippleColor="@android:color/transparent" è utile per nascondere l'animazione ripple che non seguirebbe le indicazioni del Material design 3

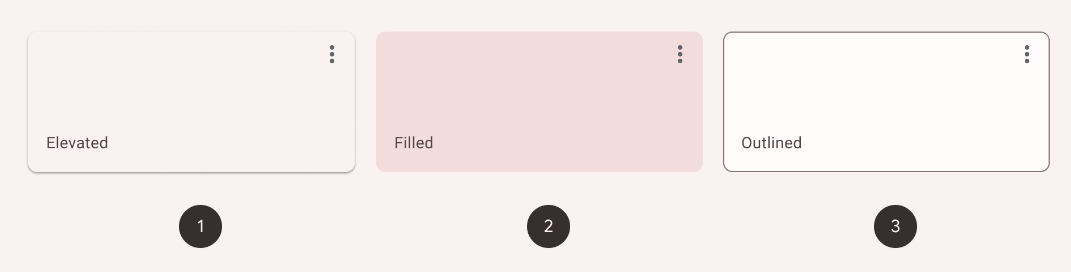
Gli elementi della Navigation bar sono descritti nel file *activity_main_drawer.xml* presente nella cartella *menu* delle risorse.
Questo è implementato con il seguente codice

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:showIn="navigation_view">

        <item
            android:id="@+id/nav_home"
            android:enabled="true"

            android:icon="@drawable/munu_selected"

            android:title="Diario" />
        <item
            android:id="@+id/nav_calc"
            android:enabled="true"

            android:icon="@drawable/calc_selected"
            android:title="Cal. Calorie" />

        <item
            android:id="@+id/nav_recipes"
            android:enabled="true"

            android:icon="@drawable/recipie_selected"
            android:title="Ricettario" />
        <item
            android:id="@+id/nav_fragment_grafici"
            android:enabled="true"

            android:icon="@drawable/grafici_selected"
            android:title="Grafici" />

        <item
            android:id="@+id/nav_user"
            android:enabled="true"

            android:icon="@drawable/user_selected"
            android:title="Utente" />
</menu>
```

2. I *Button* sono implementati in maniera classica, ma ad essi viene applicato lo stile a seconda del tipo di bottone.
   Questi sono quelli utilizzate nel nostro codice : *elevated Botton, filled Button, text button* e *text icon button.*
   Questo è un esempio base di come tutti i bottoni sono stati implementati

```
<Button
    android:id="@+id/elevatedButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Elevated button"
    style="@style/Widget.Material3.Button.ElevatedButton"
     />
```

   in particolare questa riga
   `style="@style/Widget.Material3.Button.ElevatedButton"`
   rende il bottone di tipo *elevated*



3. Le Card in Materia Design 3 sono di 3 tipi, tutti presenti all'interno di "DiarioM3", essi sono: *Elevated card, Filled card* ed *Outlined card*
   Queste Card sono state implementate nel seguente modo:

   nel quale `style` qualifica ancora una volta il tipo di Card.

```
<com.google.android.material.card.MaterialCardView

    style="@style/Widget.Material3.CardView.Outlined"
    ></com.google.android.material.card.MaterialCardView>
```

4. I Text fields che sono presenti in "DiarioM3" sono tutti di tipo *Outlined text field* e li abbiamo implementati grazie al seguente codice

```
<com.google.android.material.textfield.TextInputLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="@string/nome_utente"
    >

    <com.google.android.material.textfield.TextInputEditText

        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</com.google.android.material.textfield.TextInputLayout>
```

BIBLIOGRAFIA:

Material Design:

- *https://material.io/*

Material Design 3:

- https://m3.material.io/
- https://codelabs.developers.google.com/codelabs/apply-dynamic-color#3
- https://www.uxpin.com/studio/blog/an-introduction-to-interactions-with-material-design/
- https://www.youtube.com/c/MaterialDesign/videos
- https://www.designware.io/blog/material-design-3
- https://codelabs.developers.google.com/codelabs/mdc-101-java#0

Algoritmi per I colori dinamici:

- https://github.com/material-foundation/material-color-utilities
- https://material.io/blog/science-of-color-design
- https://arxiv.org/abs/1101.0395

DiarioM3

- https://alimentazionesportiva.it/formula-fabbisogno-calorico/