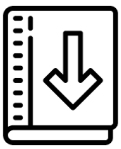
MODUL 12
Styles dan themes, material design, dimens dan colors

CAPAIAN PEMBELAJARAN

Mahasiswa dapat membuat desain aplikasi sederhana secara mandiri.

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. Android Studio 3.4.
2. Handphone Android versi 7.0 (Nougat)
3. Kabel data USB.
4. Driver ADB.



DASAR TEORI

Apa itu Material Design dan Material Component untuk Android?

Material Design adalah sistem untuk membangun produk digital yang berani dan indah. Dengan menyatukan gaya, branding, interaksi, dan gerak di bawah seperangkat prinsip dan komponen yang konsisten, tim produk dapat mewujudkan potensi desain terbesar mereka.

Untuk aplikasi Android, **Material Components for Android (MDC Android**) menyatukan desain dan rekayasa dengan library komponen untuk menciptakan konsistensi di seluruh aplikasi. Ketika sistem Desain Bahan berkembang, komponen-komponen ini diperbarui untuk memastikan implementasi pixel-konsisten yang konsisten dan kepatuhan terhadap standar pengembangan front-end Google. MDC juga tersedia untuk web, iOS, dan Flutter.

Material Components for Android (MDC Android).
MDC TextField. Fitur MDC Text Field meliputi:

- Menampilkan built-in error feedback
- Mendukung toggle untuk visibilitas password menggunakan app: passwordToggleEnabled
- Menawarkan built-in helper text functionality menggunakan app: helperText
- Menampilkan jumlah karakter total dan maks menggunakan app: counterEnabled dan app: counterMaxLength

MDC Button. Fitur MDC Button meliputi:

- Built-in touch feedback (disebut MDC Ripple) secara defauls
- Default elevation
- Customizable corner radius dan stroke

AppBarLayout.
AppBarLayout adalah LinearLayout vertikal yang mengimplementasikan banyak fitur konsep desain material app bar, yaitu scrolling gestures. Children harus menyediakan perilaku pengguliran yang diinginkan melalui setScrollFlags (int) dan atribut layout xml terkait: app: layout_scrollFlags. Pandangan ini sangat bergantung pada apa yang digunakan sebagai anak langsung dalam CoordinatorLayout. Jika kita menggunakan AppBarLayout di dalam ViewGroup yang berbeda, sebagian besar fungsinya tidak akan berfungsi. AppBarLayout juga membutuhkan saudara scrolling yang terpisah untuk mengetahui kapan harus scrolling. Pengikatan dilakukan melalui kelas perilaku AppBarLayout.ScrollingViewBehavior, yang berarti bahwa kita harus mengatur perilaku tampilan scrolling menjadi turunan dari AppBarLayout.ScrollingViewBehavior. Resource string yang berisi nama kelas lengkap tersedia.

MaterialCardView.
Kelas ini memasok gaya Material untuk card dalam konstruktor. Widget akan menampilkan style Material default yang benar tanpa menggunakan style flag. Lebar Stroke dapat diatur menggunakan atribut strokeWidth. Atur warna stroke menggunakan atribut strokeColor. Tanpa strokeColor, kartu tidak akan memberikan batas garis, terlepas dari nilai strokeWidth. Card mengimplementasikan Checkable, cara default untuk beralih ke android: checked_state tidak disediakan. Klien harus memanggil setChecked (boolean). Ini menunjukkan aplikasi: checkedIcon dan mengubah warna overlay. Card juga memiliki status khusus yang dimaksudkan untuk digunakan ketika card adalah aplikasi yang dapat diseret: dragged_state. Ini digunakan dengan memanggil setDragged (boolean). Ini mengubah warna overlay dan mengangkat kartu untuk menyampaikan gerakan. Catatan: Hindari pengaturan setClipToOutline (boolean) menjadi true. Ada tampilan perantara untuk klip konten, pengaturan ini akan memiliki konsekuensi kinerja negatif. Hirarki tampilan aktual yang ada di bawah MaterialCardView TIDAK dijamin untuk cocok dengan hierarki tampilan yang ditulis dalam XML. Akibatnya, panggilan ke getParent () pada anak-anak dari MaterialCardView, tidak akan mengembalikan MaterialCardView itu sendiri, melainkan tampilan perantara. Jika kita perlu mengakses MaterialCardView secara langsung, atur android: id dan gunakan findViewById (int).

PRAKTIK

https://codelabs.developers.google.com/codelabs/mdc-101-kotlin/#0

1. Kita akan membuat aplikasi yang memanfaatkan material design.
2. Untuk Project awal, gunakan project yang sudah ada, kemudian jalankan. Anda akan menemukan tampilan sebagai berikut.

3. Selanjutnya kita akan menambahkan textfield. Kita akan menggunakan komponen textfield MDC, yang mencakup fungsi bawaan yang menampilkan floating label and error message.
4. Buka file shr_login_fragment.xml dari resource layout, kemudian tambahkan dua elemen TextInputLayout dengan child TextInputEditText di dalam <LinearLayout>, di bawah label "SHRINE" <TextView>

```
<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="4dp"
    android:hint="@string/shr_hint_username">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/password_text_input"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="4dp"
    android:hint="@string/shr_hint_password">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/password_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>
```

5. Tambahkan validasi input. Komponen TextInputLayout menyediakan fungsionalitas umpan balik kesalahan bawaan. Untuk menampilkan umpan balik kesalahan, lakukan perubahan berikut ke shr_login_fragment.xml:

   Setel atribut app:errorEnabled menjadi true pada elemen Password di TextInputLayout. Ini akan menambahkan extra padding untuk pesan kesalahan di bawah text field.

   Setel atribut android: inputType ke "textPassword" pada elemen Password TextInputEditText. Ini akan menyembunyikan teks input di password field.

```
<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="4dp"
    android:hint="@string/shr_hint_username">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/password_text_input"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="4dp"
    android:hint="@string/shr_hint_password"
    app:errorEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/password_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword" />
</com.google.android.material.textfield.TextInputLayout>
```

6. Jalankan dan amati hasilnya. Perhatikan animasi floating label.

7. Kemudian kita akan menambahkan dua buah button untuk halaman login : “Cancel” dan “Next”. Kita akan menggunakan komponen Button MDC, yang dilengkapi dengan iconic Material Design ink ripple effect built-in.
8. Buka file shr_login_fragment.xml, tambahkan <RelativeLayout> di dalam tag <LinearLayout>, dibawa elemen TextInputLayout. Kemudian tambahkan dua buah elemen <MaterialButton> di dalam tag <RelativeLayout>.

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

    <com.google.android.material.button.MaterialButton
        android:id="@+id/next_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:text="@string/shr_button_next" />

    <com.google.android.material.button.MaterialButton
        android:id="@+id/cancel_button"
        style="@style/Widget.MaterialComponents.Button.TextButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="12dp"
        android:layout_marginRight="12dp"
        android:layout_toStartOf="@id/next_button"
        android:layout_toLeftOf="@id/next_button"
        android:text="@string/shr_button_cancel" />

</RelativeLayout>
```

9. Jalankan dan amati hasilnya.
10. Kita akan menambahkan navigasi ke Fragment berikutnya. Kita akan menambahkan beberapa kode Kotlin ke LoginFragment.kt untuk menghubungkan tombol "NEXT" kita untuk transisi ke fragmen lain. Tambahkan metode private boolean isPasswordValid di LoginFragment.kt di bawah onCreateView (), dengan logika untuk menentukan apakah kata sandi itu valid atau tidak.

```
private fun isPasswordValid(text: Editable?): Boolean {
    return text != null && text.length >= 8
}
```

11. Selanjutnya kita akan menambahkan listener pada Button Next, yang menetapkan dan menghapus kesalahan berdasarkan method isPasswordValid () yang baru saja kita buat. Di onCreateView (), klik listener ini harus ditempatkan di antara inflater line dan return view line. Sekarang mari kita tambahkan key listener kunci ke password TextInputEditText untuk listen ke key events yang akan menghapus kesalahan. Listener ini juga harus menggunakan isPasswordValid() untuk memeriksa apakah password itu valid atau tidak. Tambahkan langsung di bawah click listener di onCreateView(). Method onCreateView menjadi sebagai berikut.

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View?
{
    // Inflate the layout for this fragment.
    val view = inflater.inflate(R.layout.shr_login_fragment, container, false)

    // Set an error if the password is less than 8 characters.
    view.next_button.setOnClickListener({
        if (!isPasswordValid(password_edit_text.text!!)) {
            password_text_input.error = getString(R.string.shr_error_password)
        } else {
            // Clear the error.
            password_text_input.error = null
```

```
            // Navigate to the next Fragment.
            (activity as NavigationHost).navigateTo(ProductGridFragment(), false)
        }
    })

    // Clear the error once more than 8 characters are typed.
    view.password_edit_text.setOnKeyListener({ _, _, _ ->
        if (isPasswordValid(password_edit_text.text!!)) {
            // Clear the error.
            password_text_input.error = null
        }
        false
    })
    return view
}
```

12. Jalankan dan amati hasilnya.
13. Aplikasi akan kita kembangkan dengan menambahkan komponen lainnya.
    Menambahkan top app bar. *Kita akan menambahkan navigasi untuk membantu user mengarahkan aplikasi ke lokasi yang diharapkan. Navigasi mengacu pada komponen, interaksi, isyarat visual, dan arsitektur informasi yang memungkinkan pengguna untuk bergerak seputar aplikasi. Ini membantu membuat konten dan fitur dapat ditemukan, sehingga tugas mudah diselesaikan. Material Design menawarkan pola navigasi yang memastikan tingkat kegunaan yang tinggi. Salah satu komponen yang paling terlihat adalah top app bar. Untuk menyediakan navigasi dan memberi pengguna akses cepat ke tindakan lain, kita akan tambahkan top app bar.*
14. Menambahkan widget AppBar. Buka file shr_product_grid_fragment.xml, hapus blok <LinearLayout> yang berisi "You did it!" TextView dan ganti dengan yang berikut ini:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ProductGridFragment">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/app_bar"
            style="@style/Widget.Shrine.Toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            app:title="@string/shr_app_name"
            app:navigationIcon="@drawable/shr_menu"/>
    </com.google.android.material.appbar.AppBarLayout>
</FrameLayout>
```

15. Menambahkan action buttons dan style di top app bar. Di fungsi onCreateView dari kelas ProductGridFragment.kt, atur activity Toolbar yang akan digunakan sebagai ActionBar menggunakan setSupportActionBar. Kita dapat melakukan ini setelah tampilan dibuat dengan inflater.

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View?
{
    // Inflate the layout for this fragment with the ProductGrid theme
    val view = inflater.inflate(R.layout.shr_product_grid_fragment, container, false)

    // Set up the toolbar.
    (activity as AppCompatActivity).setSupportActionBar(view.app_bar)

    return view;
}
```

16. Selanjutnya, langsung di bawah metode yang baru saja kita ubah untuk mengatur toolbar, mari kita ganti onCreateOptionsMenu untuk mengembang konten shr_toolbar_menu.xml ke dalam toolbar:

```
override fun onCreateOptionsMenu(menu: Menu, menuInflater: MenuInflater) {
    menuInflater.inflate(R.menu.shr_toolbar_menu, menu)
    super.onCreateOptionsMenu(menu, menuInflater)
}
```

17. Akhirnya, ganti onCreate () di ProductGridFragment.kt, dan setelah memanggil super (), panggil setHasOptionMenu dengan true:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setHasOptionsMenu(true)
}
```

18. Cuplikan kode di atas mengatur app bar dari layout XML menjadi Action Bar untuk activity ini. Panggilan balik padaCreateOptionsMenu memberi tahu aktivitas apa yang harus digunakan sebagai menu. Dalam hal ini, panggilan itu akan menempatkan item menu dari R.menu.shr_toolbar_menu ke app bar. File menu berisi dua item: "Search" dan "Filter".
19. Jalankan dan amati hasilnya. Sekarang toolbar memiliki ikon navigasi, judul, dan dua ikon tindakan di sisi kanan. toolbar juga menampilkan ketinggian menggunakan bayangan halus yang menunjukkan layer itu pada layer yang berbeda dari konten.
20. Menambahkan card. Kita akan menambahkan satu card di bawah top app bar. Card harus memiliki wilayah untuk gambar, judul, dan label untuk teks sekunder. Tambahkan yang berikut ini di shr_product_grid_fragment.xml di bawah AppBarLayout.

```
<com.google.android.material.card.MaterialCardView
    android:layout_width="160dp"
    android:layout_height="180dp"
    android:layout_marginBottom="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginTop="70dp"
    app:cardBackgroundColor="?attr/colorPrimaryDark"
    app:cardCornerRadius="4dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:background="#FFFFFF"
        android:orientation="vertical"
        android:padding="8dp">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="2dp"
            android:text="@string/shr_product_title"
            android:textAppearance="?attr/textAppearanceHeadline6" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:padding="2dp"
            android:text="@string/shr_product_description"
            android:textAppearance="?attr/textAppearanceBody2" />
    </LinearLayout>
</com.google.android.material.card.MaterialCardView>
```

21. Jalankan dan amati hasilnya. Kita dapat melihat Card sisipan dari tepi kiri, dan sudut-sudutnya membulat dan bayangan (yang menyatakan ketinggian Card). Seluruh elemen disebut " container." Selain dari container, semua elemen di dalamnya adalah opsional. Kita dapat menambahkan elemen-elemen berikut ke dalam container: teks header, thumbnail atau avatar, teks subjudul, pembagi, dan bahkan tombol dan ikon. Card yang baru saja kita buat, misalnya, berisi dua TextViews (satu untuk judul, dan satu untuk teks sekunder) di LinearLayout, selaras dengan bagian bawah Card. Card biasanya ditampilkan dalam koleksi dengan Card lain. Pada lanjutan project ini, kita akan meletakkannya sebagai koleksi di grid.
22. Buat grid card. Ketika beberapa card hadir di layar, card tersebut dikelompokkan bersama menjadi satu atau lebih koleksi. Card dalam grid adalah coplanar, artinya mereka berbagi resting elevation yang sama satu sama lain (kecuali diambil atau diseret).
23. Buat grid card. Ketika beberapa card hadir di layar, card tersebut dikelompokkan bersama menjadi satu atau lebih koleksi. Card dalam grid adalah coplanar, artinya mereka berbagi resting elevation yang sama satu sama lain (kecuali diambil atau diseret).
24. Siapkan grid card. Lihat dan pelajari file shr_product_card.xml yang sudah ada. Layour card ini berisi card dengan gambar (dalam hal ini, NetworkImageView, yang memungkinkan kita memuat dan menampilkan gambar dari URL), dan dua TextView. Selanjutnya, lihat dan pelajari juga file ProductCardRecyclerViewAdapter yang sudah ada. Ada dalam paket yang sama dengan ProductGridFragment. Kelas adaptor di atas mengelola konten dari grid yang ada. Untuk menentukan apa yang harus dilakukan oleh setiap tampilan dengan kontennya, kita akan segera menulis kode untuk onBindViewHolder (). Dalam paket yang sama, kita juga dapat melihat ProductCardViewHolder. Kelas ini menyimpan tampilan yang memengaruhi layout card, sehingga kita dapat memodifikasinya nanti. Untuk menyiapkan grid, pertama-tama kita menghapus placeCard MaterialCardView dari shr_product_grid_fragment.xml. Selanjutnya, kita harus menambahkan komponen yang mewakili grid card kita. Dalam hal ini, kita akan menggunakan RecyclerView. Tambahkan komponen RecyclerView ke shr_product_grid_fragment.xml kita di bawah komponen XML AppBarLayout.

```
<androidx.core.widget.NestedScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="56dp"
    android:background="@color/productGridBackgroundColor"
    android:paddingStart="@dimen/shr_product_grid_spacing"
    android:paddingEnd="@dimen/shr_product_grid_spacing"

app:layout_behavior="@string/appbar_scrolling_view_behavior">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.core.widget.NestedScrollView>
```

25. Terakhir, di onCreateView (), tambahkan kode inisialisasi RecyclerView ke ProductGridFragment.kt setelah kita memanggil setUpToolbar (tampilan) dan sebelum pernyataan return. Cuplikan kode di atas berisi langkah-langkah inisialisasi yang diperlukan untuk menyiapkan RecyclerView. Ini termasuk pengaturan layout manager RecyclerView, ditambah inisialisasi dan pengaturan adapter RecyclerView.

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View?
{
    // Inflate the layout for this fragment with the ProductGrid theme
    val view = inflater.inflate(R.layout.shr_product_grid_fragment, container, false)

    // Set up the toolbar.
    (activity as AppCompatActivity).setSupportActionBar(view.app_bar)

    // Set up the RecyclerView
    view.recycler_view.setHasFixedSize(true)
    view.recycler_view.layoutManager =
```

```
GridLayoutManager(context,2,RecyclerView.VERTICAL, false)
    val adapter = ProductCardRecyclerViewAdapter(
        ProductEntry.initProductEntryList(resources))
    view.recycler_view.adapter = adapter
    val largePadding = resources.getDimensionPixelSize(R.dimen.shr_product_grid_spacing)
    val smallPadding =
resources.getDimensionPixelSize(R.dimen.shr_product_grid_spacing_small)
    view.recycler_view.addItemDecoration(ProductGridItemDecoration(largePadding,
smallPadding))

    return view;
}
```

26. Jalankan dan amati hasilnya.
27. Menambahkan gambar dan teks. Untuk setiap card, tambahkan gambar, nama produk, dan harga. Abstraksi ViewHolder kita menampung view untuk setiap card. Di ViewHolder kita, tambahkan tiga tampilan sebagai berikut.

```
class ProductCardViewHolder(itemView: View) //TODO: Find and store views from
itemView
    : RecyclerView.ViewHolder(itemView){
    var productImage: NetworkImageView =
itemView.findViewById(R.id.product_image)
    var productTitle: TextView = itemView.findViewById(R.id.product_title)
    var productPrice: TextView = itemView.findViewById(R.id.product_price)
}
```

28. Perbarui method onBindViewHolder () di ProductCardRecyclerViewAdapter untuk menetapkan judul, harga, dan gambar produk untuk setiap tampilan produk seperti yang ditunjukkan di bawah ini:

```
override fun onBindViewHolder(holder: ProductCardViewHolder, position:
Int) {
    // TODO: Put ViewHolder binding code here in MDC-102
    if (position < productList.size) {
        val product = productList[position]
        holder.productTitle.text = product.title
        holder.productPrice.text = product.price
        ImageRequester.setImageFromUrl(holder.productImage,
product.url)
    }
}
```

29. Kode di atas memberi tahu adapter RecyclerView apa yang harus dilakukan dengan setiap card, menggunakan ViewHolder. Di sini, adapter menetapkan data teks pada masing-masing TextViews ViewHolder, dan memanggil ImageRequester untuk mendapatkan gambar dari URL. ImageRequester adalah kelas yang kita sediakan untuk kenyamanan kita, dan kelas ini menggunakan library Volley.
30. Jalankan dan amati hasilnya.

LATIHAN

1. Modifikasilah aplikasi dengan menambahkan/mengubah warna, typography, elevation dan layout.
2. Gunakan referensi berikut.

https://codelabs.developers.google.com/codelabs/mdc-103-kotlin/#0

TUGAS

1. Modifikasi aplikasi dengan mengembangkan project diatas.
2. Gunakan referensi berikut.

   https://codelabs.developers.google.com/codelabs/mdc-104-kotlin/#0

REFERENSI

1. https://kotlinlang.org/docs/reference/
2. https://developer.android.com/kotlin
3. https://developer.android.com/courses/kotlin-android-fundamentals/toc
4. https://codelabs.developers.google.com/android-kotlin-fundamentals/
5. https://developer.android.com/kotlin/learn
6. https://developer.android.com/kotlin/resources
7. https://codelabs.developers.google.com/codelabs/mdc-101-kotlin/#0
8. https://codelabs.developers.google.com/codelabs/mdc-102-kotlin/#0
9. https://codelabs.developers.google.com/codelabs/mdc-103-kotlin/#0
10. https://codelabs.developers.google.com/codelabs/mdc-104-kotlin/#0