LAPORAN PRAKTIKUM LAB D1
PENGEMBANGAN BERBASIS PLATFORM
"Android Studio Introduction"

DISUSUN OLEH:
RESMA ADI NUGROHO
24060121120021

DEPARTEMEN INFORMATIKA
FAKULTAS SAINS DAN MATEMATIKA
UNIVERSITAS DIPONEGORO
2023

BAB I

PENDAHULUAN

1.1 Tujuan

a. Mahasiswa mampu mengoperasikan Android Studio.
b. Mahasiswa mampu membuat sebuah aplikasi penghitung volume balok.

1.2 Rumusan Masalah

a. Tambahkan fitur untuk menghitung luas permukaan dan keliling balok pada project aplikasi yang sudah dikerjakan pada latihan sebelumnya. Tampilkan hasil perhitungan pada tv_result.

BAB II

DASAR TEORI

2.1 XML

File Extensible Markup Language (XML) adalah dokumen berbasis teks yang dapat Anda simpan dengan ekstensi .xml. Anda dapat menulis XML mirip dengan file teks lainnya. Untuk membuat atau mengedit file XML, Anda dapat menggunakan salah satu hal berikut:

- Editor teks seperti Notepad atau Notepad++
- Editor XML online
- Web browser

Setiap file XML menyertakan komponen berikut.

a. Dokumen XML

Tanda <xml></xml> digunakan untuk menandai awal dan akhir dari sebuah file XML. Konten dalam tanda ini juga disebut dokumen XML. Itu adalah tanda pertama yang akan dicari oleh perangkat lunak apa pun untuk memproses kode XML.

b. Deklarasi XML

Sebuah dokumen XML dimulai dengan beberapa informasi tentang XML itu sendiri. Misalnya, dokumen itu mungkin menyebutkan versi XML yang diikutinya. Pembukaan ini disebut deklarasi XML. Berikut adalah contohnya.

<?xml version="1.0" encoding="UTF-8"?>

c. Elemen XML

Semua tanda lain yang Anda buat dalam dokumen XML disebut dengan elemen XML. Elemen XML dapat berisi fitur berikut:

- Teks
- Atribut
- Elemen lainnya

Semua dokumen XML dimulai dengan tanda primer, yang disebut dengan elemen root. Contohnya seperti pada kode dibawah ini:

```
<InvitationList>
<family>
    <aunt>
    <name>Christine</name>
     <name>Stephanie</name>
    </aunt>
</family>
</InvitationList>
```

<InvitationList> adalah elemen root; family (keluarga) dan aunt (bibi) adalah nama elemen lainnya.

d. Atribut XML

Elemen XML dapat memiliki deskriptor lain yang disebut atribut. Anda dapat menentukan nama atribut Anda sendiri dan menulis nilai atribut dalam tanda kutip seperti yang ditunjukkan di bawah.

```
<person age="22">
```

e. Konten XML

Data dalam file XML juga disebut konten XML. Misalnya, dalam file XML, Anda mungkin melihat data seperti ini.

```
<friend>
    <name>Charlie</name>
    <name>Steve</name>
</friend>
```

Nilai data Charlie dan Steve adalah kontennya.

2.2 Kotlin

Kotlin adalah bahasa pemrograman modern yang dirancang untuk menggantikan Java dalam pengembangan aplikasi Android. Bahasa ini dikembangkan oleh JetBrains dan pertama kali diperkenalkan pada tahun 2011. Kotlin menjadi populer dalam pengembangan Android karena berbagai alasan.

Salah satu keunggulan Kotlin adalah sintaksis yang lebih bersih dan ringkas. Hal ini membuat kode lebih mudah dipahami dan ditulis. Kotlin juga mendukung konsep pemrograman berorientasi objek (OOP), yang memungkinkan pengembang untuk mengorganisir kode ke dalam objek-objek yang mewakili entitas nyata. Selain itu, Kotlin memiliki banyak fitur canggih seperti pengelolaan nulitas yang lebih baik, ekstensi fungsi, dan lambdas, yang semuanya membuat pengembangan lebih efisien.

2.3 Android Studio

Android Studio adalah Integrated Development Environment (IDE) resmi untuk pengembangan aplikasi Android. Berbasis editor kode dan alat developer yang andal dari IntelliJ IDEA, Android Studio menawarkan lebih banyak fitur yang mampu meningkatkan produktivitas saat mem-build aplikasi Android, seperti:

- Sistem build berbasis Gradle yang fleksibel
- Emulator yang cepat dan kaya fitur
- Lingkungan terpadu tempat Anda bisa mengembangkan aplikasi untuk semua perangkat Android
- Edit Live untuk mengupdate composable di emulator dan perangkat fisik secara real time
- Template kode dan integrasi GitHub untuk membantu Anda membuat fitur aplikasi umum dan mengimpor kode sampel
- Framework dan alat pengujian yang lengkap
- Alat lint untuk merekam performa, kegunaan, kompatibilitas versi, dan masalah lainnya
- Dukungan C++ dan NDK
- Dukungan bawaan untuk Google Cloud Platform, yang memudahkan integrasi Google Cloud Messaging dan App Engine

BAB III

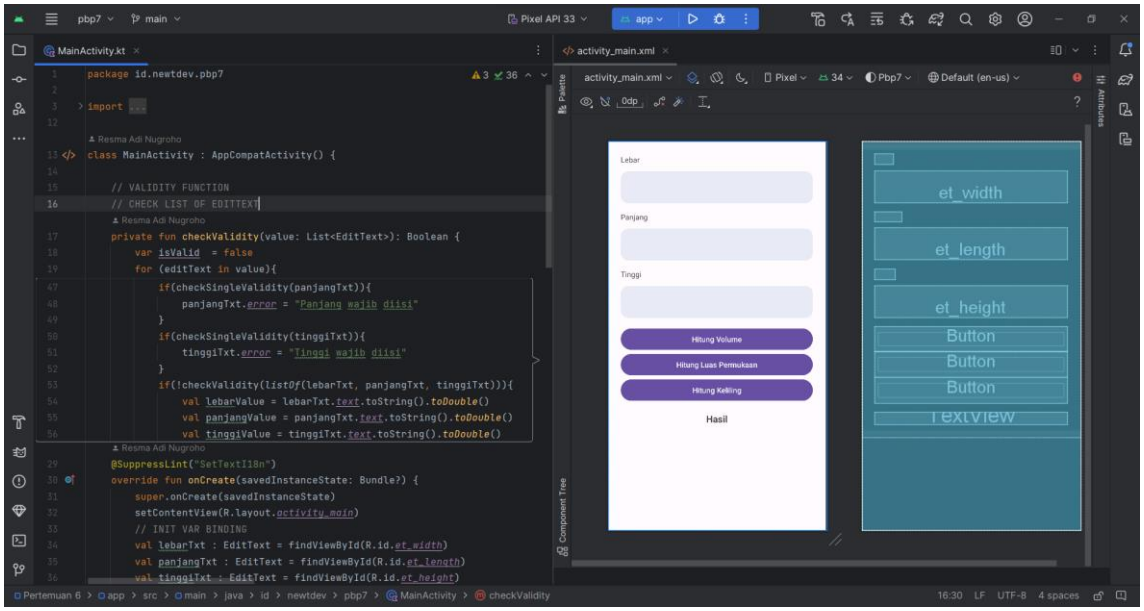
PEMBAHASAN

3.1 Aplikasi Penghitung Volume, Luas dan Keliling

Untuk membuat aplikasi berbasis android yang mampu untuk menghitung volume, luas dan keliling dapat menggunakan IDE Android Studio dengan Kotlin sebagai bahasa pemrogramannya. Selain kotlin terdapat pula XML yang digunakan untuk membuat tampilan dari aplikasi tersebut. Untuk membuat tampilan penghitungan yang terdiri dari tiga form input dan tiga button yang akan memiliki fungsi masing-masing untuk menghitung nilai tersebut maka dapat menggunakan komponen yang telah tersedia pada Android Studio yakni:

a. EditText untuk tempat input nilai yang akan digunakan dalam kalkulasi
b. TextView untuk menampilkan hasil dari perhitungan
c. Button untuk tombol yang nantinya akan digunakan untuk memanggil fungsi perhitungan

Desain tampilan dari aplikasi penghitung tersebut akan memiliki rupa seperti dibawah ini:



Gambar 1 Tampilan Antarmuka Aplikasi

Dengan kode XML yang digunakan dapat dilihat pada kode dibawah ini:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    <LinearLayout
        android:orientation="vertical"
        android:padding="24dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="12dp"
        android:text="@string/label_lebar" />
    <EditText
        android:id="@+id/et_width"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_marginBottom="16dp"
        android:paddingHorizontal="12dp"
        android:background="@drawable/rounded_et"
        android:inputType="numberDecimal"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="12dp"
        android:text="@string/label_panjang" />
    <EditText
        android:id="@+id/et_length"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_marginBottom="16dp"
        android:paddingHorizontal="12dp"
        android:background="@drawable/rounded_et"
        android:inputType="numberDecimal" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="12dp"
        android:text="@string/label_tinggi" />
    <EditText
        android:id="@+id/et_height"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_marginBottom="16dp"
        android:paddingHorizontal="12dp"
        android:background="@drawable/rounded_et"
        android:inputType="numberDecimal" />
```

```
    <Button
        android:id="@+id/btn_calculate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/label_button" />
        <Button
            android:id="@+id/btn_calculate_luas"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/label_luas" />
        <Button
            android:id="@+id/btn_calculate_keliling"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/label_keliling" />
        <TextView
            android:id="@+id/tv_result"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="18dp"
            android:text="@string/label_hasil"
            android:textAlignment="center"
            android:textSize="18sp"
            android:textStyle="bold" />
    </LinearLayout>
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Layout yang digunakan untuk pada tampilan tersebut menggunakan ConstraintLayout, sehingga elemen yang ada pada layout tersebut akan menyesuaikan dengan constraint yang ditentukan. Kemudian terdapat juga untuk element ScrollView yang berguna agar tampilan layout dapat discroll. Kemudian pada bagian dalam ScrollView terdapat element LinearLayout dengan orientation ‘vertical’ sehingga element pada LinearLayout tersebut akan terletakkan secara vertical. Kemudian di dalam element LinearLayout akan terdiri element-element TextView, EditText dan Button yang akan membentuk tampilan seperti pada gambar sebelumnya. Untuk masing-masing element tersebut akan memiliki atribut yang dapat digunakan untuk memodifikasi dan mengubah tampilan element tersebut. Contohnya pada EditText terdapat atribut background yang menggunakan background rounded_et.xml yang akan menghasilkan tampilan rounded dan berwarna pada EditText tersebut.

```
<?xml version="1.0" encoding="utf-8"?>
<!--  res/drawable/rounded_edittext.xml -->
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle"
    android:padding="10dp">

    <solid android:color="#E8EAF6" />
    <corners
        android:bottomRightRadius="15dp"
        android:bottomLeftRadius="15dp"
        android:topLeftRadius="15dp"
        android:topRightRadius="15dp" />
</shape>
```

Selain itu untuk masing-masing element juga perlu memiliki id yang nantinya akan digunakan pada binding pada kode kotlin untuk activity yang terkait. Setelah tampilan telah selesai dibuat maka langkah selanjutnya membuat bagian logic activity pada file kotlin.

Pada file kotlin untuk activity akan berisi beberapa method yang perlu digunakan contohnya onCreate. Selain itu terdapat pula override method yang dapat digunakan seperti onDestroy, onResume, dan beberapa method lain yang dapat dimanfaatkan dan dioptimalkan kegunaannya. Method-method tersebut merupakan serangkaian pada Lifecycle Activity pada android.

Pada bagian awal onCreate, element yang sebelumnya sudah pernah dibuat perlu dilakukan binding terlebih dahulu menggunakan findViewById, selain itu sebenarnya kita dapat menggunakan metode lain yakni dengan ActiviyBinding sehingga proses binding tidak perlu kita lakukan karena telah terbind secara otomatis, sehingga apabila element tersebut perlu digunakan dapat dipanggil dengan binding.nama_element. Namun pada kasus ini akan digunakan binding secara konvensional menggunakan findViewById.

```
// INIT VAR BINDING
val LebarTxt : EditText = findViewById(R.id.et_width)
val PanjangTxt : EditText = findViewById(R.id.et_length)
val TinggiTxt : EditText = findViewById(R.id.et_height)
val calculateBtn : Button = findViewById(R.id.btn_calculate)
val luasBtn : Button = findViewById(R.id.btn_calculate_luas)
val kelBtn : Button = findViewById(R.id.btn_calculate_keliling)
val resultView : TextView = findViewById(R.id.tv_result)
val formData : Map<String, EditText> = mapOf(
    "Lebar" to LebarTxt,
    "Panjang" to PanjangTxt,
    "Tinggi" to TinggiTxt
)
```

Gambar 2 Inisialisasi Variabel dan Binding

Perlu diperhatikan juga ketika akan membuat sebuah variable dapat diperhatikan untuk tipe data yang digunakan. Setelah element berhasil dibinding maka selanjutnya dapat dibuat sebuah validasi agar form tidak bisa disubmit sebelum semua input diisi. Karena akan terdapat tiga button yang akan memiliki validasi yang sama namun dengan perhitungan yang berbeda maka validasi akan dibuat sedikit modular sehingga fungsi akan bisa digunakan berulang. Validasi akan dibuat dengan multi check validation dan multi error checking.

```
private fun multiCheck(value: Map<String, EditText>): Boolean {
    var isValid  = false
    value.map { isValid = it.value.text.isEmpty() || (it.value.text.isBlank()) }
    return isValid
}
// MULTI ERROR SHOW
 Resma Adi Nugroho *
private fun multiError(value: Map<String, EditText>) {
    value.map { it: Map.Entry<String, EditText>
        if(it.value.text.isEmpty()) {
            it.value.error = "${it.key} harus diisi"
        }
    }
}
```

Gambar 3 Fungsi Validasi

Penggunaan kode modular seperti ini dimungkinkan ketika terdapat perubahan atau penambahan fitur lagi akan memudahkan dalam proses pengembangan sebuah aplikasi.

```
// CHECK FORM VALIDATION
fun checkFormValidation(form: Map<String, EditText>, callback: (MutableMap<String, Double>) -> Unit) {
    multiError(form)
    if(!multiCheck(form)){
        var data = mutableMapOf<String, Double>()
        form.map { data.put(it.key, it.value.text.toString().toDouble()) }
        callback(data)
        Toast.makeText( context: this,  text: "Berhasil menghitung", Toast.LENGTH_SHORT).show()
    }
}
```

Gambar 4 Fungsi Validasi Form

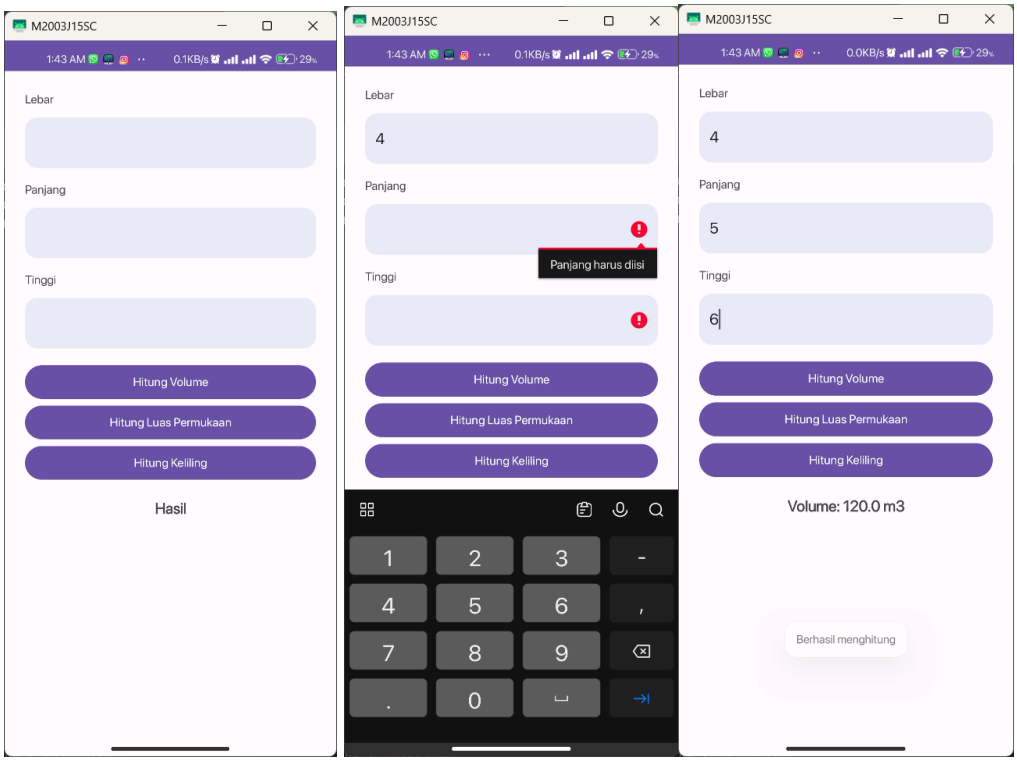
Selanjutnya untuk fungsi validasi form dapat dibuat dengan fungsi atau method yang sebelumnya sudah didefinisikan. Sehingga kode akan seperti diatas. Check form akan memiliki dua parameter yakni form bertipe Map<String, EditText> dan callback berupa Function param. Pertama akan dicek untuk form apakah terdapat input yang kosong dengan multiError. Apabila true maka akan menampilkan pesan error pada input tersebut. Apabila semua input sudah benar maka akan masuk ke proses perhitungan untuk masing-masing buttonya. Data yang akan dikalkulasi akan didapatkan dari variable form yang akan dimapping kembali ke dalam sebuah MutableMap atau map yang dapat diubah. Setelah itu nilai data akan dikembalikan dengan callback function dan akan menampilkan pesan sukses setelah callback function.

```
// CALCULATE FUNCTION
calculateBtn.setOnClickListener { it: View!
    checkFormValidation(formData) { it: MutableMap<String, Double>
        val volumeCount = it.getValue( key: "Lebar") * it.getValue( key: "Panjang") * it.getValue( key: "Tinggi")
        resultView.text = "Volume: $volumeCount m3"
    }
}
luasBtn.setOnClickListener { it: View!
    checkFormValidation(formData) { it: MutableMap<String, Double>
        val luasCount = 2 * ((it.getValue( key: "Panjang")*it.getValue( key: "Lebar")) + (it.getValue( key: "Panjang")
        resultView.text = "Luas: $luasCount m2"
    }
}
kelBtn.setOnClickListener { it: View!
    checkFormValidation(formData) { it: MutableMap<String, Double>
        val kelCount = 4 *(it.getValue( key: "Lebar") + it.getValue( key: "Panjang") * it.getValue( key: "Tinggi"))
        resultView.text = "Keliling: $kelCount m"
    }
}
```

Gambar 5 Fungsi Perhitungan

Fungsi validasi form dapat dipanggil untuk masing-masing button yang telah dibuat sebelumnya. Dengan memanggil setOnClickListener maka button akan dapat menerima action apabila ditekan. Fungsi check form validation memiliki parameter sebuah function yang apabila diimplementasikan maka dapat menggunakan tanda {} secara langsung setelah tanda tutup fungsi berakhir. Karena callback function bertipe Map maka untuk mengakses datanya dapat menggunakan funsgi getValue dengan mengarahkan ke kunci tertentu. Setelah data didapatkan maka dapat dilakukan kalkulasi berdasarkan rumus yang telah dibuat sebelumnya. Kemudian hasil dapat ditampilkan pada resultView melalui atribut text dari resultView. Text ditampilkan dengan literal method agar nilai variable dapat digabungkan langsung dengan text.

3.2 Hasil Aplikasi



Gambar 6 Hasil Run Aplikasi

BAB IV

PENUTUP

Android Studio dan Kotlin adalah dua komponen kunci dalam pengembangan aplikasi Android. Android Studio adalah lingkungan pengembangan yang diberikan oleh Google untuk mempermudah pengembangan aplikasi Android. Ini mencakup alat untuk menyusun kode, menguji aplikasi, dan merancang antarmuka pengguna.

Kotlin, di sisi lain, adalah bahasa pemrograman yang dapat digunakan dalam pengembangan aplikasi Android. Dengan Kotlin, pengembang dapat menulis kode yang lebih bersih, efisien, dan aman. Ini merupakan alternatif yang lebih baik daripada bahasa Java yang lebih lama.

Selain itu, XML (eXtensible Markup Language) adalah bahasa yang digunakan untuk mendefinisikan tata letak antarmuka pengguna dalam aplikasi Android. Dalam XML, kita dapat menentukan di mana elemen-elemen seperti tombol, teks, dan gambar harus ditempatkan dalam aplikasi. Ini memisahkan antarmuka pengguna dari logika aplikasi, yang memudahkan pengembangan dan pemeliharaan aplikasi.

Dengan menggabungkan Android Studio, Kotlin, dan XML, pengembang memiliki alat yang kuat untuk menciptakan aplikasi Android yang menarik dan efisien. Mereka dapat merancang antarmuka pengguna dengan mudah, menulis kode dengan lebih sedikit kerumitan, dan menguji aplikasi mereka sepanjang proses. Kombinasi dari ketiganya merupakan fondasi kuat untuk menciptakan aplikasi Android yang sukses.

DAFTAR PUSTAKA

AWS (Amazon Web Services). "XML - Extensible Markup Language." Diakses dari: https://aws.amazon.com/id/what-is/xml/.

Android Developers. "Android Studio: The Official IDE for Android." Diakses dari: https://developer.android.com/studio/intro?hl=id.