Materi Detail Input Form dan State Management Jetpack Compose

Pengantar

Dalam pengembangan aplikasi Android modern menggunakan Jetpack Compose, pengelolaan input dan perubahan data merupakan bagian yang sangat penting. Jetpack Compose menggunakan pendekatan deklaratif sehingga tampilan akan otomatis berubah ketika data berubah.

Konsep utama yang digunakan adalah:

1. TextField
2. State
3. remember
4. mutableStateOf
5. State Hoisting

Materi ini membahas secara detail bagaimana membuat form input data pada aplikasi Android menggunakan Jetpack Compose.

---

1. TextField

Pengertian TextField

TextField adalah komponen input pada Jetpack Compose yang digunakan untuk menerima masukan teks dari pengguna.

TextField mirip seperti EditText pada Android XML tradisional.

Struktur Dasar TextField

```
TextField(
    value = "",
    onValueChange = {}
)
```

Penjelasan Parameter

Parameter	Fungsi
value	Nilai teks saat ini
onValueChange	Event ketika teks berubah

Contoh Sederhana

```
@Composable
fun SimpleTextField() {

    var nama by remember {
        mutableStateOf("")
    }

    TextField(
        value = nama,
        onValueChange = {
            nama = it
        },
        label = {
            Text("Nama")
        }
    )
}
```

Penjelasan Alur

1. User mengetik teks.
2. onValueChange dipanggil.
3. Nilai state berubah.
4. Compose melakukan recomposition.
5. UI otomatis diperbarui.

---

2. State pada Jetpack Compose

Pengertian State

State adalah data yang dapat berubah selama aplikasi berjalan.

Compose akan mengamati perubahan state dan memperbarui tampilan secara otomatis.

Contoh State

```
var nama = "Budi"
```

Tetapi variabel biasa tidak cukup untuk Compose.

Compose membutuhkan observable state.

Observable State

```
var nama by remember {
    mutableStateOf("")
}
```

Kenapa State Penting?

Karena UI pada Compose bersifat:

- Reactive
- Declarative
- Dynamic

Ketika state berubah:

- UI otomatis berubah
- Tidak perlu findViewById
- Tidak perlu notifyDataSetChanged

---

3. remember

Pengertian remember

remember digunakan untuk menyimpan state selama composable masih aktif.

Tanpa remember, data akan kembali ke nilai awal setiap recomposition.

Contoh Tanpa remember

```
@Composable
fun Counter() {
```

```
    var angka = 0

    Button(onClick = {
        angka++
    }) {
        Text("$angka")
    }
}
```

Hasil:

Nilai selalu kembali ke 0.

Contoh Dengan remember

```
@Composable
fun Counter() {

    var angka by remember {
        mutableStateOf(0)
    }

    Button(onClick = {
        angka++
    }) {
        Text("$angka")
    }
}
```

Sekarang nilai akan tersimpan.

Fungsi remember

Fungsi	Keterangan
Menyimpan state	Agar tidak hilang saat recomposition
Optimasi UI	Menghindari reset data
Reactive UI	Memungkinkan UI update otomatis

---

4. mutableStateOf

Pengertian mutableStateOf

mutableStateOf digunakan untuk membuat state yang dapat diamati Compose.

Sintaks

```
mutableStateOf(nilai_awal)
```

Contoh

```
var nama by remember {
    mutableStateOf("")
}
```

Cara Kerja

Ketika nilai berubah:

```
nama = "Andi"
```

Compose otomatis:

- mendeteksi perubahan
- menjalankan recomposition
- memperbarui UI

Perbedaan Variabel Biasa dan mutableStateOf

Variabel Biasa	mutableStateOf
Tidak reactive	Reactive
UI tidak update	UI otomatis update
Tidak diamati Compose	Diamati Compose

---

5. State Hoisting

Pengertian State Hoisting

State Hoisting adalah teknik memindahkan state keluar composable.

Tujuannya:

- Reusable component
- Clean architecture
- Mudah testing
- Separation of concern

Tanpa State Hoisting

```
@Composable
fun NameInput() {

    var nama by remember {
        mutableStateOf("")
    }

    TextField(
        value = nama,
        onValueChange = {
            nama = it
        }
    )
}
```

Masalah:

- State terkunci di composable
- Sulit digunakan ulang

Dengan State Hoisting

```
@Composable
fun NameInput(
    nama: String,
    onNamaChange: (String) -> Unit
) {
    TextField(
        value = nama,
        onValueChange = onNamaChange
    )
}
```

Parent:

```
@Composable
fun MainScreen() {
```

```
    var nama by remember {
        mutableStateOf("")
    }

    NameInput(
        nama = nama,
        onNamaChange = {
            nama = it
        }
    )
}
```

Keuntungan State Hoisting

Keuntungan	Penjelasan
Reusable	Component bisa dipakai ulang
Clean Code	Pemisahan logic dan UI
Mudah Testing	State dapat diuji
Scalability	Mudah dikembangkan

Praktik: Form Input Data

Studi Kasus

Aplikasi Registrasi Siswa

Aplikasi digunakan untuk:

- Input nama siswa
- Input email siswa
- Validasi sederhana
- Menampilkan hasil registrasi

Tampilan Aplikasi

Komponen:

1. TextField Nama
2. TextField Email

3. Tombol Daftar
4. Pesan validasi

Source Code Lengkap

```
package com.example.registrasisiswa

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            RegistrationScreen()
        }
    }
}

@Composable
fun RegistrationScreen() {

    var nama by remember {
        mutableStateOf("")
    }

    var email by remember {
        mutableStateOf("")
    }

    var pesan by remember {
        mutableStateOf("")
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
```

```
) {

    Text(
        text = "Registrasi Siswa",
        style = MaterialTheme.typography.headlineMedium
    )

    Spacer(modifier = Modifier.height(20.dp))

    TextField(
        value = nama,
        onValueChange = {
            nama = it
        },
        label = {
            Text("Nama")
        },
        modifier = Modifier.fillMaxWidth()
    )

    Spacer(modifier = Modifier.height(12.dp))

    TextField(
        value = email,
        onValueChange = {
            email = it
        },
        label = {
            Text("Email")
        },
        modifier = Modifier.fillMaxWidth()
    )

    Spacer(modifier = Modifier.height(20.dp))

    Button(
        onClick = {

            if (nama.isEmpty() || email.isEmpty()) {
                pesan = "Semua field harus diisi"
            }
            else if (!email.contains("@")) {
                pesan = "Format email tidak valid"
            }
            else {
                pesan = "Registrasi berhasil"
            }
        }
    ) {
```

```
            Text("Daftar")
        }

        Spacer(modifier = Modifier.height(20.dp))

        Text(text = pesan)
    }
}
```

Penjelasan Kode

1. State Nama

```
var nama by remember {
    mutableStateOf("")
}
```

Digunakan untuk menyimpan input nama.

---

2. State Email

```
var email by remember {
    mutableStateOf("")
}
```

Digunakan untuk menyimpan input email.

---

3. State Pesan

```
var pesan by remember {
    mutableStateOf("")
}
```

Digunakan untuk menampilkan pesan validasi.

---

4. TextField Nama

```
TextField(
    value = nama,
```

```
    onValueChange = {
        nama = it
    }
)
```

Ketika user mengetik:

- nama berubah
- UI update otomatis

---

5. Validasi

```
if (nama.isEmpty() || email.isEmpty())
```

Memastikan semua field diisi.

---

6. Validasi Email

```
else if (!email.contains("@"))
```

Memastikan email valid.

Output Aplikasi

Kondisi 1

Field kosong.

Output:

```
Semua field harus diisi
```

Kondisi 2

Email salah.

Output:

```
Format email tidak valid
```

Kondisi 3

Data benar.

Output:

```
Registrasi berhasil
```

Alur Kerja Compose pada Form

```
User Input
     ↓
TextField
     ↓
onValueChange
     ↓
State berubah
     ↓
Compose mendeteksi perubahan
     ↓
Recomposition
     ↓
UI diperbarui otomatis
```

Kesimpulan

TextField

Digunakan untuk menerima input user.

State

Digunakan untuk menyimpan data yang berubah.

remember

Menyimpan state selama composable aktif.

mutableStateOf

Membuat state reactive.

State Hoisting

Memindahkan state keluar composable agar reusable.

Latihan Mahasiswa

Latihan 1

Tambahkan field:

- Nomor HP
- Alamat

Latihan 2

Tambahkan validasi:

- Panjang password minimal 8 karakter
- Email harus valid

Latihan 3

Buat form login menggunakan:

- Email
- Password
- Tombol Login

Challenge Project

Buat aplikasi:

Student Registration App

Fitur:

- Input nama
- Input email
- Input jurusan
- Validasi form
- Menampilkan data registrasi
- Reset form

Gunakan:

- Jetpack Compose
- State Management
- State Hoisting
- Material 3