LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

BUILD A SCROLLABLE LIST

Oleh:

Ghani Mudzakir **NIM. 2310817110011**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2024

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 3

Laporan Praktikum Pemrograman Mobile Modul 3: Build a Scrollable List ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Prakitkum ini dikerjakan oleh:

Nama Praktikan : Ghani Mudzakir
NIM : 2310817110011

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar NIM. 2210817210026	Muti`a Maulida S.Kom M.T.I NIP. 19881027 201903 20 13

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Soal Praktikum:

1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:

    1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)

    2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas

    3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah

    4. Terdapat 2 button dalam list, dengan fungsi berikut:

    a. Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain

    b. Button kedua menggunakan Navigation component/intent untuk membuka laman detail item

    5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius

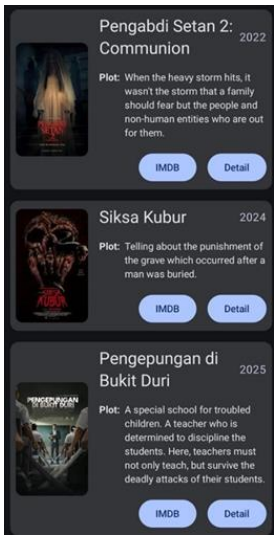
    6. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang

    7. Aplikasi menggunakan arsitektur single activity (satu activity memiliki beberapa fragment)

    8. Aplikasi berbasis XML harus menggunakan ViewBinding

2. Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

UI item list harus berisi 1 gambar, 2 button (intent eksplisit dan navigasi), dan 2 baris teks dan setiap baris memiliki 2 teks yang berbeda. Diusahakan agar desain UI item list menyerupai UI berikut:



Gambar 1 Contoh UI List

Desain UI laman detail bebas, tetapi diusahakan untuk mengikuti kaidah desain Material Design dan data item ditampilkan penuh di laman detail seperti contoh berikut:

Gambar 2 Contoh UI Detail

A. Source Code

1. Pembukaan.kt

Tabel 1 Source Code Pembukaan.kt

```
1  package com.example.pembukaan_catur
2
```

```
3  import androidx.annotation.DrawableRes
4
5  data class TampilanPembukaan(
6      val nama_pembukaan: String,
7      val penjelasan_singkat: String,
8      val penjelasan_dua_paragraf: String,
9      @DrawableRes val imageResId: Int,
10     val link: String
11 )
```

2. PembukaanRepository.kt

Tabel 2 Source Code PembukaanRepository.kt

```
1  package com.example.pembukaan_catur
2
3  class PembukaanRepository {
4      fun ambilSemuaPembukaan(): List<TampilanPembukaan>{
5          return listOf(
6              TampilanPembukaan(
7                  nama_pembukaan = "Sicilian Defense",
8                  penjelasan_singkat = "Sebuah pembukaan agresif
9  yang dimulai dengan e4 c5, sering digunakan untuk menciptakan
10 permainan tidak seimbang.",
11                 penjelasan_dua_paragraf = """
12         Sicilian Defense adalah salah satu pembukaan paling
13 populer dan tajam dalam catur modern. Dengan membalas e4 dengan
14 c5, hitam menghindari simetri dan berusaha mengambil inisiatif
15 dengan permainan sayap.
16
17         Pembukaan ini terkenal menghasilkan posisi yang kompleks
18 dan penuh taktik. Banyak juara dunia, seperti Kasparov dan
19 Fischer, mengandalkan Sicilian untuk menghadapi pemain e4.
20     """.trimIndent(),
21                 imageResId = R.drawable.p1,
22                 link = "https://www.chess.com/openings/Sicilian-
23 Defense"
24             ),
25
26             TampilanPembukaan(
27                 nama_pembukaan = "French Defense",
28                 penjelasan_singkat = "Strategi yang dimulai
29 dengan e4 e6, di mana hitam memblokir pion putih dan merencanakan
30 serangan melalui posisi yang lebih tertutup.",
31                 penjelasan_dua_paragraf = """
32         French Defense memberikan struktur pion yang kokoh dan
33 solid untuk hitam. Ia menawarkan banyak kemungkinan strategis,
34 terutama dalam permainan tengah.
35
```

```
        Salah satu ciri khasnya adalah konflik antara pion e5 putih dan struktur pertahanan hitam di d5. Taktik dan strategi memainkan peran penting dalam membuka posisi ini.
    """.trimIndent(),
                imageResId = R.drawable.p2,
                link  =  "https://www.chess.com/openings/French-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Ruy López Opening",
                penjelasan_singkat  =  "Pembukaan  klasik  yang dimulai dengan e4 e5 2.Nf3 Nc6 3.Bb5, bertujuan untuk mengontrol pusat dan menekan pertahanan hitam.",
                penjelasan_dua_paragraf = """
        Ruy López adalah salah satu pembukaan tertua yang masih dimainkan di level tinggi. Dengan menekan kuda di c6, putih mencoba mengganggu kontrol hitam terhadap pusat.

        Posisi yang timbul sering bersifat strategis, dengan ruang untuk manuver jangka panjang dan potensi serangan raja di tahap akhir pembukaan.
    """.trimIndent(),
                imageResId = R.drawable.p3,
                link   =   "https://www.chess.com/openings/Ruy-Lopez-Opening"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Caro-Kann Defense",
                penjelasan_singkat = "Dimulai dengan e4 c6, hitam berusaha  untuk  memperkuat  pusat  dan  membangun  pertahanan  yang solid.",
                penjelasan_dua_paragraf = """
        Caro-Kann terkenal karena stabilitas dan keamanan bagi raja hitam. Ini adalah pembukaan pilihan bagi pemain yang menyukai posisi bertahan namun aktif.

        Hitam  membentuk  d5  segera  setelah  c6,  berusaha menetralisir pusat putih tanpa menciptakan kelemahan signifikan.
    """.trimIndent(),
                imageResId = R.drawable.p4,
                link   =   "https://www.chess.com/openings/Caro-Kann-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Italian Game",
```

```
                penjelasan_singkat = "Sebuah pembukaan yang
sering mengarah pada permainan terbuka, dimulai dengan e4 e5 2.Nf3
Nc6 3.Bc4, dengan tujuan menyerang pusat lawan.",
                penjelasan_dua_paragraf = """
        Italian Game memberikan peluang pengembangan cepat bagi
kedua pihak. Putih langsung mengincar titik lemah f7, titik lemah
paling rentan bagi hitam di awal permainan.

        Pembukaan ini cocok untuk pemain pemula hingga master
karena keseimbangan antara taktik dan strategi.
    """.trimIndent(),
                imageResId = R.drawable.p5,
                link = "https://www.chess.com/openings/Italian-
Game"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Queen's Gambit",
                penjelasan_singkat = "Pembukaan populer yang
dimulai dengan d4 d5 2.c4, di mana putih menawarkan pion untuk
mengontrol pusat papan.",
                penjelasan_dua_paragraf = """
        Queen's Gambit adalah salah satu pembukaan tertua dan
paling dihormati dalam catur. Meski disebut 'gambit', pion yang
dikorbankan biasanya dapat direbut kembali.

        Tujuan utama putih adalah mengalihkan pion d5 hitam dan
menciptakan dominasi penuh di pusat papan. Banyak juara dunia
telah menggunakan pembukaan ini dengan sukses besar.
    """.trimIndent(),
                imageResId = R.drawable.p6,
                link = "https://www.chess.com/openings/Queens-
Gambit"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Slav Defense",
                penjelasan_singkat = "Dimulai dengan d4 d5 2.c4
c6, hitam bertujuan untuk menjaga pusat dan bersiap untuk melawan
serangan putih.",
                penjelasan_dua_paragraf = """
        Slav Defense adalah respon solid terhadap Queen’s Gambit.
Dengan memainkan c6, hitam memperkuat kontrol atas d5 tanpa
membuka terlalu banyak ruang.

        Ini menghasilkan posisi yang seimbang namun fleksibel,
memungkinkan transisi ke berbagai rencana strategis tergantung
perkembangan permainan.
    """.trimIndent(),
```

```
                imageResId = R.drawable.p7,
                link   =   "https://www.chess.com/openings/Slav-
Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "King's Indian Defense",
                penjelasan_singkat  =  "Strategi  yang  dimulai
dengan 1.d4 Nf6 2.c4 g6, hitam berencana menyerang pusat dengan
pion dan pasukan yang dikembangkan setelahnya.",
                penjelasan_dua_paragraf = """
        King's Indian Defense dikenal dengan pendekatannya yang
agresif dan asimetris. Hitam mengizinkan putih membangun pusat
besar, lalu menyerangnya.

        Strategi khasnya adalah serangan raja oleh hitam, bahkan
ketika putih mengembangkan keunggulan ruang. Ini adalah pilihan
ideal bagi pecatur taktis.
    """.trimIndent(),
                imageResId = R.drawable.p8,
                link   =   "https://www.chess.com/openings/Kings-
Indian-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Nimzo-Indian Defense",
                penjelasan_singkat = "Dimulai dengan d4 Nf6 2.c4
e6 3.Nc3 Bb4, hitam bertujuan untuk mengendalikan pusat sambil
mengembangkan tekanan terhadap pion putih.",
                penjelasan_dua_paragraf = """
        Nimzo-Indian  Defense  adalah  pembukaan  strategis  yang
memanfaatkan ancaman terhadap struktur pion putih di awal. Dengan
Bb4, hitam menekan Nc3 dan menciptakan potensi kerusakan struktur.

        Pembukaan ini menggabungkan kontrol pusat dengan tekanan
posisi dan cocok bagi pemain yang menyukai fleksibilitas.
    """.trimIndent(),
                imageResId = R.drawable.p9,
                link   =   "https://www.chess.com/openings/Nimzo-
Indian-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Queen's Indian Defense",
                penjelasan_singkat  =  "Dimulai  dengan  1.d4  Nf6
2.c4 e6 3.Nf3 b6, hitam berusaha mengembangkan bidaknya dengan
cara yang fleksibel dan mengontrol pusat.",
                penjelasan_dua_paragraf = """
```

```
        Queen's Indian Defense fokus pada perkembangan bidak yang
fleksibel dan pengendalian diagonal panjang dengan gajah di b7.
Ini adalah pembukaan yang solid dan penuh manuver.

        Pembukaan ini cocok untuk pemain yang ingin menghindari
konflik langsung di awal, namun siap melawan balik saat permainan
berkembang.
    """.trimIndent(),
                imageResId = R.drawable.p10,
                link  =  "https://www.chess.com/openings/Queens-
Indian-Defense"
            )

        )
    }
}
```

3. MainActivity.kt

Tabel 3 Source Code MainActivity.kt

```
1  package com.example.pembukaan_catur
2
3  import android.content.Intent
4  import android.net.Uri
5  import android.os.Bundle
6  import androidx.activity.ComponentActivity
7  import androidx.activity.compose.setContent
8  import androidx.activity.enableEdgeToEdge
9  import androidx.compose.foundation.Image
10 import androidx.compose.foundation.background
11 import androidx.compose.foundation.layout.Arrangement
12 import androidx.compose.foundation.layout.Column
13 import androidx.compose.foundation.layout.PaddingValues
14 import androidx.compose.foundation.layout.Row
15 import androidx.compose.foundation.layout.Spacer
16 import androidx.compose.foundation.layout.WindowInsets
17 import androidx.compose.foundation.layout.asPaddingValues
18 import androidx.compose.foundation.layout.defaultMinSize
19 import androidx.compose.foundation.layout.fillMaxSize
20 import androidx.compose.foundation.layout.fillMaxWidth
21 import androidx.compose.foundation.layout.height
22 import androidx.compose.foundation.layout.padding
23 import androidx.compose.foundation.layout.size
24 import androidx.compose.foundation.layout.statusBars
25 import androidx.compose.foundation.layout.width
26 import androidx.compose.foundation.layout.wrapContentWidth
27 import androidx.compose.foundation.lazy.LazyColumn
28 import androidx.compose.foundation.lazy.items
```

```
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.Card
import androidx.compose.material3.CardDefaults
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import
androidx.compose.ui.layout.ModifierLocalBeyondBoundsLayout
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.example.pembukaan_catur.ui.theme.Pembukaan_caturTheme
import androidx.navigation.compose.NavHost
import androidx.navigation.NavHostController
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import androidx.navigation.navArgument
import kotlin.math.round

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Pembukaan_caturTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                ) {
                    val navController = rememberNavController()
                    NavHost(
                        navController = navController,
                        startDestination = "PembukaanRepository"
                    ) {
                        composable("PembukaanRepository") {
                            DaftarPembukaan(navController)
```

```
78                          }
79
80                          composable(
                                route = "penjelasan/{desc}/{img}",
                                arguments = listOf(
                                    navArgument("desc") { type =
NavType.StringType },
                                    navArgument("img") { type =
NavType.IntType }
                                )
                            ) { backStackEntry ->
                                val              desc              =
backStackEntry.arguments?.getString("desc") ?: ""
                                val               img              =
backStackEntry.arguments?.getInt("img") ?: 0
                                tampilanDetail(img = img, nama =
"Detail Pembukaan", penjelasan = desc)
                            }
                        }
                    }
                }
            }
        }
    }
}



// tampilan per entitas
@Composable
fun TampilanEntitasPembukaan(name: String, image: Int, url:
String, description: String, penjelasan: String, navController:
NavController) {
    val context = LocalContext.current
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(10.dp),
        shape = RoundedCornerShape(16.dp),
        elevation = CardDefaults.cardElevation(defaultElevation =
4.dp)
    ) {
        Row(
            modifier = Modifier
                .padding(16.dp)
                .fillMaxWidth(),
            verticalAlignment = Alignment.CenterVertically
        ) {
            Image(
                painter = painterResource(id = image),
```

```
                contentDescription = null,
                modifier = Modifier
                    .size(width = 120.dp, height = 120.dp)
            )
            Spacer(modifier = Modifier.width(16.dp))
            Column(
                modifier = Modifier
                    .weight(1f)
            ) {
                Text(text = name, fontWeight = FontWeight.Bold,
fontSize = 17.sp)
                Spacer(modifier = Modifier.height(4.dp))
                Text(
                    text = description,
                    fontSize = 12.sp,
                    maxLines = 5,
                    overflow = TextOverflow.Ellipsis
                )
                Spacer(modifier = Modifier.height(8.dp))
                Row(
                    horizontalArrangement                       =
Arrangement.SpaceBetween,
                    modifier = Modifier
                        .fillMaxWidth()
                        .wrapContentWidth(Alignment.Start),
                ) {
                    Button(
                        onClick = {
                            val           encodedDesc           =
Uri.encode(penjelasan)

navController.navigate("penjelasan/$encodedDesc/$image")
                        },
                        contentPadding                          =
PaddingValues(horizontal = 16.dp, vertical = 8.dp),
                        shape = RoundedCornerShape(50),
                        modifier                                =
Modifier.defaultMinSize(minWidth = 1.dp),
                        colors = ButtonDefaults.buttonColors(
                            containerColor = Color.Black,
                            contentColor = Color.White
                        )
                    ) {
                        Text("Detail", fontSize = 14.sp, )
                    }

                    Spacer(modifier = Modifier.width(8.dp))

                    Button(
```

```
                        onClick = {
                            val             intent              =
Intent(Intent.ACTION_VIEW, Uri.parse(url))
                            context.startActivity(intent)
                        },
                        contentPadding                              =
PaddingValues(horizontal = 16.dp, vertical = 8.dp),
                        shape = RoundedCornerShape(50),
                        modifier                                    =
Modifier.defaultMinSize(minWidth = 1.dp),
                        colors = ButtonDefaults.buttonColors(
                            containerColor = Color.Black,
                            contentColor = Color.White
                        )
                    ) {
                        Text("Website", fontSize = 13.sp)
                    }
                }
            }
        }
    }
}

// tampilan looping untuk lazycolumn
@Composable
fun DaftarPembukaan(navController: NavHostController) {

    val repositoriPembukaan = PembukaanRepository()
    val                     fetchedPembukaan                        =
repositoriPembukaan.ambilSemuaPembukaan()
    LazyColumn(
        modifier = Modifier
            .fillMaxWidth()
            .padding(20.dp)
    ) {
        items(items=fetchedPembukaan) {
            pembukaan -> TampilanEntitasPembukaan(
                name = pembukaan.nama_pembukaan,
                image = pembukaan.imageResId,
                url = pembukaan.link,
                description = pembukaan.penjelasan_singkat,
                penjelasan = pembukaan.penjelasan_dua_paragraf,
                navController = navController
            )
        }
    }
}

@Composable
```

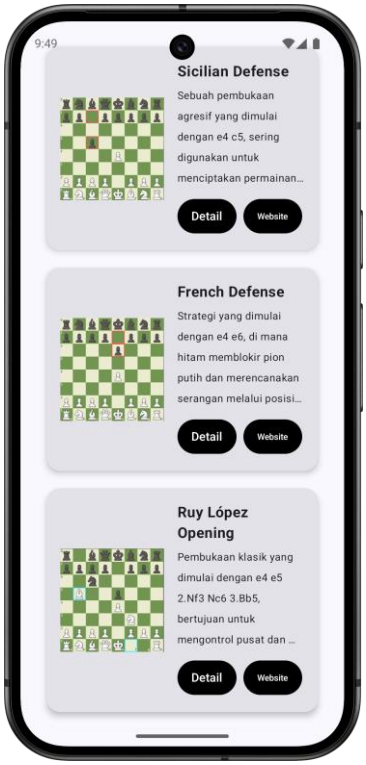
```
fun tampilanDetail(img: Int, nama: String, penjelasan: String) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
            .padding(WindowInsets.statusBars.asPaddingValues()),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Image(
            painter = painterResource(id = img),
            contentDescription = "Gambar Detail dari $nama",
            modifier = Modifier
                .fillMaxWidth()
                .height(380.dp)
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(
            text = nama,
            fontSize = 30.sp,
            fontWeight = FontWeight.W800
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(
            text = penjelasan,
            fontSize = 20.sp,
        )
    }
}



@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    Pembukaan_caturTheme {
        tampilanDetail(R.drawable.p1, "Sisilia Defense", """
        Sicilian Defense adalah salah satu pembukaan paling populer dan tajam dalam catur modern. Dengan membalas e4 dengan c5, hitam menghindari simetri dan berusaha mengambil inisiatif dengan permainan sayap.

        Pembukaan ini terkenal menghasilkan posisi yang kompleks dan penuh taktik. Banyak juara dunia, seperti Kasparov dan Fischer, mengandalkan Sicilian untuk menghadapi pemain e4.
    """.trimIndent())
    }
}
```

B. Output Program



Gambar 3 Tampilan Awal UI Aplikasi

Gambar 4 Tampilan Halaman Detail

Gambar 5 Tampilan Saat Menuju Website

C. Pembahasan

1. Pembukaan.kt

Pada file ini kita sebenarnya hanya membuat sebuah class dengan nama TampilanPembukaan dengan beberapa val parameter penting seperti nama_pembukaan dengan tipe data string, penjelasan_singkat dengan tipe data string, penjelasan_dua_paragraf dengan tipe data string, val dengan tipe data Drawable yang memiliki default value Int, dan yang terakhir adalah link dengan tipe data String. File ini akan digunakan data awal yang akan ditampilkan di bagian looping sesuai dengan MainActivity.kt nantinya.

2. PembukaanRepository.kt

Pada file ini kita membuat class dengan nama ambilSemuaPembukaan yang mengambil fungsi TampilanPembukaan dari file Pembukaan.kt dan akan mengembalikan fungsi itu dengan data-data yang dideklarasikan di file ini di dalam MainAcivity.kt

3. MainActivity.kt

Pada file MainActivity.kt ini seluruh komponen utama dari aplikasi kita didefinisikan dan dideklarasikan. Mulai dari tampilan daftar pembukaan, komponen per item, hingga tampilan detail. Pertama-tama, terdapat kelas MainActivity yang akan dieksekusi saat aplikasi pertama kali dijalankan. Pada fungsi onCreate, kita memanggil fungsi enableEdgeToEdge() untuk mengaktifkan tampilan layar secara penuh. Selanjutnya, komponen setContent digunakan untuk membungkus seluruh isi tampilan dengan tema Pembukaan_caturTheme. Kita juga memanggil Surface sebagai pembungkus utama dan NavHost digunakan untuk menangani navigasi antar halaman, dengan rute awal "PembukaanRepository".

Navigasi terdiri dari dua rute utama yaitu "PembukaanRepository" untuk menampilkan daftar pembukaan dan "penjelasan/{desc}/{img}" untuk menampilkan detail dari masing-masing pembukaan. Data seperti deskripsi dan gambar dikirim melalui argumen menggunakan fungsi Uri.encode() agar dapat terbaca dengan benar di URL. Saat rute "penjelasan/{desc}/{img}" dijalankan, data akan diteruskan ke

fungsi tampilanDetail() yang akan menampilkan gambar, nama, dan penjelasan lengkap yang berisi variabel penjelasan_dua_paragraf dari masing-masing pembukaan catur yang dipilih.

Komponen DaftarPembukaan berfungsi untuk menampilkan daftar seluruh pembukaan catur. Pada DaftarPembukaan, kita memanggil repository PembukaanRepository yang menyediakan data pembukaan. Kita menggunakan komponen LazyColumn untuk menampilkan data secara scrollable. Tiap item di dalamnya akan diteruskan ke fungsi TampilanEntitasPembukaan, yang bertugas untuk menampilkan satu entitas pembukaan catur lengkap dengan nama, gambar, penjelasan singkat, dan dua tombol sesuai dengan tampilan yang kita buat di fungsi TampilanEntitasPembukaan.

Fungsi TampilanEntitasPembukaan menampilkan komponen berbentuk Card, yang berisi gambar pembukaan di sebelah kiri dan informasi teks di sebelah kanan. Terdapat dua tombol yaitu “Detail” untuk menavigasi ke halaman penjelasan dan tombol “Website” yang membuka link ke sumber referensi yaitu chess.com menggunakan Intent.

Terakhir, fungsi tampilanDetail akan digunakan untuk menampilkan penjelasan lebih lengkap terkait satu pembukaan catur. Gambar akan ditampilkan di bagian atas, dilanjutkan dengan nama dan penjelasan dalam dua paragraf.

2. RecyclerView masih digunakan karena memberikan kontrol penuh atas perilaku dan performa daftar seperti pengelolaan tampilan yang banyak serta efisiensi memori yang baik, hal ini menjadi sangat penting untuk aplikasi kita apabila aplikasi kita sudah berskala besar atau dengan data dinamis. Meskipun kode RecyclerView cenderung boiler-plate, yang berarti fleksibilitasnya lebih tinggi dibandingkan *LazyColumn* di Jetpack Compose, hal ini menyebabkan bahwa RecyclerView lebih cocok untuk antarmuka deklaratif dan kebutuhan yang lebih sederhana atau modern.

D. Tautan Git

https://github.com/KunyitAlami/Praktikum-Pemrograman-Mobile-I-Ghani-Mudzakir.git