Navigasi

Praktikum Pemrograman Mobile - 09

Wanda Gusdya

2023

Jetpack Compose Navigation

Komponen Navigation Compose menyediakan dukungan untuk aplikasi Jetpack Compose. Kita dapat menavigasi antar composable sekaligus memanfaatkan infrastruktur dan fitur komponen Navigasi.

Latihan

Untuk menggunakan komponen navigation compose, kita harus menambahkan dependensi berikut di file build.gradle

```
//compose
implementation "androidx.navigation:navigation-compose:2.5.3"
```

Setelah itu, silakan sinkronisasikan project gradle.

Selanjutnya, tambahkan kata kunci suspend di fungsi find dalam interface SetoranSampahDao.

```
@Query("SELECT * FROM SetoranSampah WHERE id = :id")
suspend fun find(id: String): SetoranSampah?
```

Sehingga, kelas SetoranSampahDao akan terlihat seperti berikut.

```
package id.ac.unpas.functionalcompose.persistences

import androidx.lifecycle.LiveData
import androidx.room.*
import id.ac.unpas.functionalcompose.model.SetoranSampah

@Dao
interface SetoranSampahDao {
    @Query("SELECT * FROM SetoranSampah ORDER BY tanggal DESC")
    fun loadAll(): LiveData<List<SetoranSampah>>

    @Query("SELECT * FROM SetoranSampah ORDER BY tanggal DESC")
    suspend fun getList(): List<SetoranSampah>

    @Query("SELECT * FROM SetoranSampah WHERE id = :id")
    suspend fun find(id: String): SetoranSampah?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAll(vararg items: SetoranSampah)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAll(items: List<SetoranSampah>)

    @Delete
```

```
    fun delete(item: SetoranSampah)

    @Query("DELETE FROM SetoranSampah WHERE id = :id")
    fun delete(id: String)
}
```

Kemudian, tambahkan fungsi find dlam kelas SetoranSampahRepository dengan kode sebagai berikut.

```
suspend fun find(id: String) : SetoranSampah? {
    return dao.find(id)
}
```

Sehingga, kelas SetoranSampahRepository akan terlihat seperti berikut.

```
package id.ac.unpas.functionalcompose.repositories

import com.benasher44.uuid.uuid4
import com.skydoves.sandwich.message
import com.skydoves.sandwich.suspendOnError
import com.skydoves.sandwich.suspendOnException
import com.skydoves.sandwich.suspendOnSuccess
import com.skydoves.whatif.whatIfNotNull
import id.ac.unpas.functionalcompose.model.SetoranSampah
import id.ac.unpas.functionalcompose.networks.SetoranSampahApi
import
id.ac.unpas.functionalcompose.persistences.SetoranSampahDao
import javax.inject.Inject

class SetoranSampahRepository @Inject constructor(
    private val api: SetoranSampahApi,
    private val dao: SetoranSampahDao
) : Repository {

    suspend fun loadItems(
        onSuccess: (List<SetoranSampah>) -> Unit,
        onError: (List<SetoranSampah>, String) -> Unit
    ) {
        val list: List<SetoranSampah> = dao.getList()
        api.all()
            // handle the case when the API request gets a
success response.
            .suspendOnSuccess {
                data.whatIfNotNull {
                    it.data?.let { list ->
                        dao.insertAll(list)
                        val items: List<SetoranSampah> =
dao.getList()
```

```
                        onSuccess(items)
                    }
                }
            }
            // handle the case when the API request gets an
error response.
            // e.g. internal server error.
            .suspendOnError {
                onError(list, message())
            }
            // handle the case when the API request gets an
exception response.
            // e.g. network connection error.
            .suspendOnException {
                onError(list, message())
            }

    }

    suspend fun insert(
        tanggal: String,
        nama: String,
        berat: String,
        onSuccess: (SetoranSampah) -> Unit,
        onError: (SetoranSampah?, String) -> Unit
    ) {
        val id = uuid4().toString()
        val item = SetoranSampah(id, tanggal, nama, berat)
        dao.insertAll(item)
        api.insert(item)
            // handle the case when the API request gets a
success response.
            .suspendOnSuccess {
                onSuccess(item)
            }
            // handle the case when the API request gets an
error response.
            // e.g. internal server error.
            .suspendOnError {
                onError(item, message())
            }
            // handle the case when the API request gets an
exception response.
            // e.g. network connection error.
            .suspendOnException {
                onError(item, message())
            }
```

```
    }

    suspend fun update(
        id: String,
        tanggal: String,
        nama: String,
        berat: String,
        onSuccess: (SetoranSampah) -> Unit,
        onError: (SetoranSampah?, String) -> Unit
    ) {
        val item = SetoranSampah(id, tanggal, nama, berat)
        dao.insertAll(item)
        api.update(id, item)
            // handle the case when the API request gets a
success response.
            .suspendOnSuccess {
                onSuccess(item)
            }
            // handle the case when the API request gets an
error response.
            // e.g. internal server error.
            .suspendOnError {
                onError(item, message())
            }
            // handle the case when the API request gets an
exception response.
            // e.g. network connection error.
            .suspendOnException {
                onError(item, message())
            }

    }

    suspend fun delete(id: String, onSuccess: () -> Unit,
onError: (String) -> Unit) {
        dao.delete(id)
        api.delete(id)
            // handle the case when the API request gets a
success response.
            .suspendOnSuccess {
                data.whatIfNotNull {
                    onSuccess()
                }
            }
            // handle the case when the API request gets an
error response.
```

```
            // e.g. internal server error.
            .suspendOnError {
                onError(message())
            }
            // handle the case when the API request gets an
exception response.
            // e.g. network connection error.
            .suspendOnException {
                onError(message())
            }
    }

    suspend fun find(id: String) : SetoranSampah? {
        return dao.find(id)
    }
}
```

Gunakan fungsi find dalam kelas ViewModel, tambahkan juga fungsi untuk melakukan update data SetoranSampah.

```
suspend fun loadItem(id: String, onSuccess: (SetoranSampah?) ->
Unit) {
    val item = setoranSampahRepository.find(id)
    onSuccess(item)
}

suspend fun update(id: String,
                   tanggal: String,
                   nama: String,
                   berat: String){
    _isLoading.postValue(true)
    setoranSampahRepository.update(id, tanggal, nama, berat,
onError = { item, message ->
        _toast.postValue(message)
        _isLoading.postValue(false)
    }, onSuccess = {
        _isLoading.postValue(false)
        _success.postValue(true)
    })
}
```

Sehingga, kelas PengelolaanSampahViewModel akan terlihat seperti berikut.

```
package id.ac.unpas.functionalcompose.screens

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
```

```
import androidx.lifecycle.ViewModel
import dagger.hilt.android.lifecycle.HiltViewModel
import id.ac.unpas.functionalcompose.model.SetoranSampah
import
id.ac.unpas.functionalcompose.repositories.SetoranSampahReposito
ry
import javax.inject.Inject

@HiltViewModel
class PengelolaanSampahViewModel @Inject constructor(private val
setoranSampahRepository: SetoranSampahRepository) : ViewModel()
{
    private val _isLoading: MutableLiveData<Boolean> =
MutableLiveData(false)
    val isLoading: LiveData<Boolean> get() = _isLoading

    private val _success: MutableLiveData<Boolean> =
MutableLiveData(false)
    val success: LiveData<Boolean> get() = _success

    private val _toast: MutableLiveData<String> =
MutableLiveData()
    val toast: LiveData<String> get() = _toast

    private val _list: MutableLiveData<List<SetoranSampah>> =
MutableLiveData()
    val list: LiveData<List<SetoranSampah>> get() = _list

    suspend fun loadItems()
    {
        _isLoading.postValue(true)
        setoranSampahRepository.loadItems(onSuccess = {
            _isLoading.postValue(false)
            _list.postValue(it)
        }, onError = { list, message ->
            _toast.postValue(message)
            _isLoading.postValue(false)
            _list.postValue(list)
        })
    }

    suspend fun insert(tanggal: String,
                       nama: String,
                       berat: String){
        _isLoading.postValue(true)
        setoranSampahRepository.insert(tanggal, nama, berat,
onError = { item, message ->
```

```
            _toast.postValue(message)
            _isLoading.postValue(false)
        }, onSuccess = {
            _isLoading.postValue(false)
            _success.postValue(true)
        })
    }

    suspend fun loadItem(id: String, onSuccess: (SetoranSampah?)
-> Unit) {
        val item = setoranSampahRepository.find(id)
        onSuccess(item)
    }

    suspend fun update(id: String,
                       tanggal: String,
                       nama: String,
                       berat: String){
        _isLoading.postValue(true)
        setoranSampahRepository.update(id, tanggal, nama, berat,
onError = { item, message ->
            _toast.postValue(message)
            _isLoading.postValue(false)
        }, onSuccess = {
            _isLoading.postValue(false)
            _success.postValue(true)
        })
    }
}
```

Ubah nama FormPencatatanSampah menjadi FormPencatatanSampahScreen, karena kita akan memisahkannya dari PengelolaanSampahScreen. Kemudian, tambahkan parameter-parameter navController, id, dan modifier. Parameter navController akan kita gunakan untuk melakukan navigasi ke halaman lain, parameter id akan kita gunakan untuk melakukan edit data (untuk mode insert, id akan dibiarkan null), parameter modifier akan digunakan untuk menyesuaikan padding dengan scaffold yang akan menampung semua composable.

```
fun FormPencatatanSampahScreen(navController :
NavHostController, id: String? = null, modifier: Modifier =
Modifier) {
```

Kemudian, ubah fungsi dari event onClick menjadi seperti berikut untuk menangani update data dan redirect ke halaman utama menggunakan fungsi navigate.

```
onClick = {
    if (id == null) {
        scope.launch {
```

```
            viewModel.insert(tanggal.value.text,
nama.value.text, berat.value.text)
        }
    } else {
        scope.launch {
            viewModel.update(id, tanggal.value.text,
nama.value.text, berat.value.text)
        }
    }
    navController.navigate("pengelolaan-sampah")
}
```

Lalu tambahkan kode berikut di bagian bawah fungsi FormPencatatanSampahScreen untuk menampilkan data yang telah disimpan sebelumnya.

```
if (id != null) {
    LaunchedEffect(scope) {
        viewModel.loadItem(id) { setoranSampah ->
            setoranSampah?.let {
                tanggal.value =
TextFieldValue(setoranSampah.tanggal)
                nama.value = TextFieldValue(setoranSampah.nama)
                berat.value =
TextFieldValue(setoranSampah.berat)
            }
        }
    }
}
```

Sekarang, fungsi FormPencatatanSampahScreen akan terlihat seperti berikut.

```
package id.ac.unpas.functionalcompose.screens

import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.Button
import androidx.compose.material.ButtonDefaults
import androidx.compose.material.OutlinedTextField
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.rememberCoroutineScope
```

```
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalLifecycleOwner
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.input.KeyboardCapitalization
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.TextFieldValue
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import androidx.navigation.NavHostController
import id.ac.unpas.functionalcompose.ui.theme.Purple700
import id.ac.unpas.functionalcompose.ui.theme.Teal200
import kotlinx.coroutines.launch

@Composable
fun FormPencatatanSampahScreen(navController :
NavHostController, id: String? = null, modifier: Modifier =
Modifier) {
    val viewModel = hiltViewModel<PengelolaanSampahViewModel>()

    val tanggal = remember
{ mutableStateOf(TextFieldValue("")) }
    val nama = remember { mutableStateOf(TextFieldValue("")) }
    val berat = remember { mutableStateOf(TextFieldValue("")) }

    val isLoading = remember { mutableStateOf(false) }
    val buttonLabel = if (isLoading.value) "Mohon tunggu..."
else "Simpan"

    val scope = rememberCoroutineScope()

    Column(modifier = modifier
        .fillMaxWidth()) {

        OutlinedTextField(
            label = { Text(text = "Tanggal") },
            value = tanggal.value,
            onValueChange = {
                tanggal.value = it
            },
            modifier = Modifier
                .padding(4.dp)
                .fillMaxWidth(),
            placeholder = { Text(text = "yyyy-mm-dd") }
        )
```

```
        OutlinedTextField(
            label = { Text(text = "Nama") },
            value = nama.value,
            onValueChange = {
                nama.value = it
            },
            modifier = Modifier
                .padding(4.dp)
                .fillMaxWidth(),
            keyboardOptions = KeyboardOptions(capitalization =
KeyboardCapitalization.Characters, keyboardType =
KeyboardType.Text),
            placeholder = { Text(text = "XXXXX") }
        )

        OutlinedTextField(
            label = { Text(text = "Berat") },
            value = berat.value,
            onValueChange = {
                berat.value = it
            },
            modifier = Modifier
                .padding(4.dp)
                .fillMaxWidth(),
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Decimal),
            placeholder = { Text(text = "5") }
        )

        val loginButtonColors = ButtonDefaults.buttonColors(
            backgroundColor = Purple700,
            contentColor = Teal200
        )

        val resetButtonColors = ButtonDefaults.buttonColors(
            backgroundColor = Teal200,
            contentColor = Purple700
        )

        Row (modifier = Modifier
            .padding(4.dp)
            .fillMaxWidth()) {
            Button(modifier = Modifier.weight(5f), onClick = {
                if (id == null) {
                    scope.launch {
                        viewModel.insert(tanggal.value.text,
nama.value.text, berat.value.text)
```

```
                    }
                } else {
                    scope.launch {
                        viewModel.update(id, tanggal.value.text,
nama.value.text, berat.value.text)
                    }
                }
                navController.navigate("pengelolaan-sampah")
            }, colors = loginButtonColors) {
                Text(
                    text = buttonLabel,
                    style = TextStyle(
                        color = Color.White,
                        fontSize = 18.sp
                    ), modifier = Modifier.padding(8.dp)
                )
            }

            Button(modifier = Modifier.weight(5f), onClick = {
                tanggal.value = TextFieldValue("")
                nama.value = TextFieldValue("")
                berat.value = TextFieldValue("")
            }, colors = resetButtonColors) {
                Text(
                    text = "Reset",
                    style = TextStyle(
                        color = Color.White,
                        fontSize = 18.sp
                    ), modifier = Modifier.padding(8.dp)
                )
            }
        }
    }

    viewModel.isLoading.observe(LocalLifecycleOwner.current) {
        isLoading.value = it
    }

    if (id != null) {
        LaunchedEffect(scope) {
            viewModel.loadItem(id) { setoranSampah ->
                setoranSampah?.let {
                    tanggal.value =
TextFieldValue(setoranSampah.tanggal)
                    nama.value =
TextFieldValue(setoranSampah.nama)
                    berat.value =
```

```
TextFieldValue(setoranSampah.berat)
                    }
                }
            }
        }
}
```

Kemudian, tambahkan juga parameter navController dan modifier untuk fungsi PengelolaanSampahScreen, untuk tujuan yang sama dengan sebelumnya.

```
@Composable
fun PengelolaanSampahScreen(navController : NavHostController,
modifier: Modifier = Modifier) {
```

Ubah nilai parameter modifier pada komponen Column utama menjadi modifier (dengan m kecil) lalu hapus pemanggilan FormPencatatanSampah() di bagian atas Column utama tersebut. Tambahkan sebuah Button untuk membuka halaman untuk menambah data setoran sampah.

```
Column(modifier = modifier.fillMaxWidth()) {
    Button(onClick = {
        navController.navigate("tambah-pencatatan-sampah")
    }) {
        Text(text = "Tambah")
    }
```

Tambahkan atribut clickable pada modifier Row di dalam items, lalu isi dengan kode untuk membuka halaman edit pengelolaan sampah berdasarkan id.

```
Row(modifier = Modifier
    .padding(15.dp)
    .fillMaxWidth().clickable {
        navController.navigate("edit-pengelolaan-
sampah/${item.id}")
    }) {
```

Buatlah file MainScreen.kt di package screens, kemudian buat fungsi MainScreen seperti kode berikut. Kita menyimpan routing navigasi di dalam komponen NavHost. Perhatikan bahwa ada route yang memiliki parameter, dan ada yang tidak.

```
package id.ac.unpas.functionalcompose.screens

import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxHeight
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
```

```
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.width
import androidx.compose.material.ContentAlpha
import androidx.compose.material.Icon
import androidx.compose.material.IconButton
import androidx.compose.material.LocalContentAlpha
import androidx.compose.material.MaterialTheme
import androidx.compose.material.ProvideTextStyle
import androidx.compose.material.Scaffold
import androidx.compose.material.Snackbar
import androidx.compose.material.SnackbarHost
import androidx.compose.material.Text
import androidx.compose.material.TopAppBar
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Menu
import androidx.compose.material.rememberScaffoldState
import androidx.compose.runtime.Composable
import androidx.compose.runtime.CompositionLocalProvider
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import androidx.navigation.navArgument
import id.ac.unpas.functionalcompose.ui.theme.Purple700

@Composable
fun MainScreen() {
    val navController = rememberNavController()
    val scaffoldState = rememberScaffoldState()

    val title = remember { mutableStateOf("") }
    val appBarHorizontalPadding = 4.dp

    Scaffold(
        topBar = {
            TopAppBar(
                backgroundColor = Purple700,
                elevation = 0.dp,
                modifier= Modifier.fillMaxWidth()) {
```

```
                //TopAppBar Content
                Box(Modifier.height(32.dp)) {

                    Row(Modifier.fillMaxHeight()
                        .width(72.dp - appBarHorizontalPadding),
verticalAlignment = Alignment.CenterVertically) {
                        CompositionLocalProvider(
                            LocalContentAlpha provides
ContentAlpha.high,
                        ) {
                            IconButton(
                                onClick = {  },
                                enabled = true,
                            ) {
                                Icon(Icons.Filled.Menu, null,
tint = Color.White)
                            }
                        }
                    }


                    //Title
                    Row(Modifier.fillMaxSize(),
                        verticalAlignment =
Alignment.CenterVertically) {

                        ProvideTextStyle(value =
MaterialTheme.typography.h6) {
                            CompositionLocalProvider(
                                LocalContentAlpha provides
ContentAlpha.high,
                            ){
                                Text(
                                    modifier =
Modifier.fillMaxWidth(),
                                    textAlign =
TextAlign.Center,
                                    color = Color.White,
                                    maxLines = 1,
                                    text = title.value
                                )
                            }
                        }
                    }
                }
            }
```

```
        },
        scaffoldState = scaffoldState,
        snackbarHost = {
            // reuse default SnackbarHost to have default
animation and timing handling
            SnackbarHost(it) { data ->
                // custom snackbar with the custom colors
                Snackbar(
                    actionColor = Color.Green,
                    contentColor = Color.White,
                    snackbarData = data
                )
            }
        },
    )
    { innerPadding ->
        Column(
            modifier = Modifier.fillMaxSize(),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            NavHost(navController = navController,
startDestination = "pengelolaan-sampah") {
                composable("pengelolaan-sampah") {
                    title.value = "Pengelolaan Sampah"
                    PengelolaanSampahScreen(navController =
navController, modifier = Modifier.padding(innerPadding))
                }

                composable("tambah-pengelolaan-sampah") {
                    title.value = "Tambah Pengelolaan Sampah"
                    FormPencatatanSampahScreen(navController =
navController, modifier = Modifier.padding(innerPadding))
                }

                composable("edit-pengelolaan-sampah/{id}",
listOf(
                    navArgument("id") {
                        type = NavType.StringType
                    }
                )) { backStackEntry ->
                    title.value = "Edit Pengelolaan Sampah"
                    val id =

backStackEntry.arguments?.getString("id")
                            ?: return@composable

                    FormPencatatanSampahScreen(navController =
```

```
navController, id = id, modifier =
Modifier.padding(innerPadding))
                }

            }
        }
    }
}
```

Ubah MainActivity sehingga memanggil fungsi MainScreen di fungsi onCreate dan DefaultPreview seperti pada kode berikut.

```
Surface(
    modifier = Modifier.fillMaxSize(),
    color = MaterialTheme.colors.background
) {
    MainScreen()
}
```

```
@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    FunctionalComposeTheme {
        MainScreen()
    }
}
```

Sehingga, sekarang kelas MainActivity akan terlihat seperti berikut.

```
package id.ac.unpas.functionalcompose

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import dagger.hilt.android.AndroidEntryPoint
import id.ac.unpas.functionalcompose.screens.MainScreen
import
```

```
id.ac.unpas.functionalcompose.ui.theme.FunctionalComposeTheme

@AndroidEntryPoint
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            FunctionalComposeTheme {
                // A surface container using the 'background'
color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    MainScreen()
                }
            }
        }
    }
}

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    FunctionalComposeTheme {
        MainScreen()
    }
}
```

Jalankan dan amati hasilnya.