
Velkommen til gruppetime i IN2000!

Gruppe 1 med Ida, Jamila & Jørgen

Viktige beskjeder!

- Underkjent oblig? :(
- Dobbeltsjekk feedback og status + evt. ny frist



Oppsummering oblig 1

Det viktigste

- Navigasjon med NavHost + navController
 - Kjøring av enhetstester
 - Bruk av @Preview
 - Bruk av remember
 - LocalSoftwareKeyboardController
 - ExposedDropDownMenuBox
 - Pakker
 - Kodesplitting + kodeprinsipper
-

Figma / Material Theme

Material 3 / Figma revansj

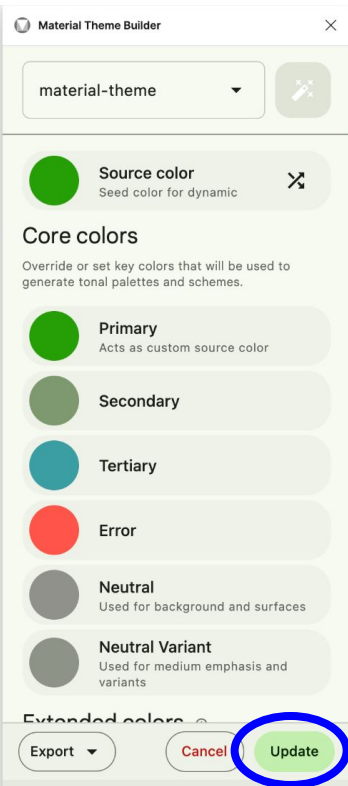
Jeg prøver meg igjen!

Nå har jeg satt meg inn i hva problemet var sist, og siden jeg ikke får deltatt i flere gruppetimer lager jeg slides som Jørgen og Jamila må klare å formulere for meg :)

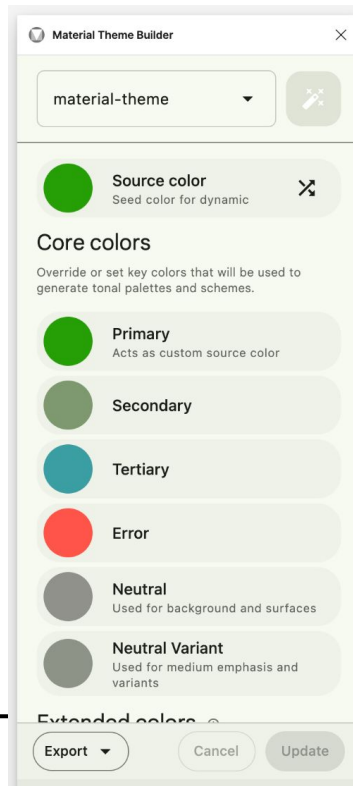
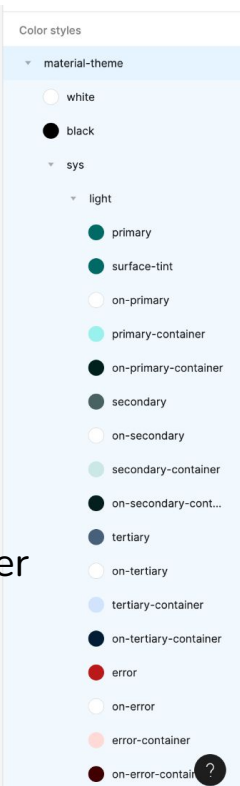
Håper dette gir mer mening, og er mer oversiktlig enn da jeg prøvde meg live. Nok en gang har jeg innsett at det sjeldent er en god idé...

Lage tema i Figma

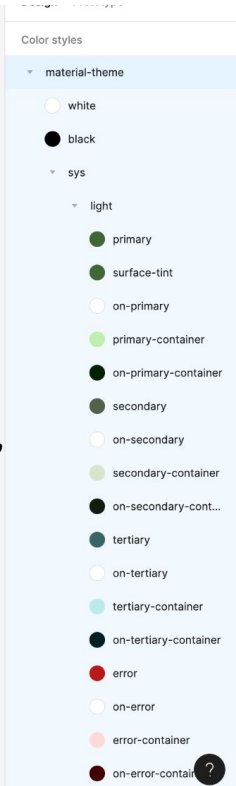
Temaet du får laget ved hjelp av “Material Theme Builder” lagres i Figma, som gjør det lett å bruke fargene når man skal designe!



Når du har valgt/fått nytt fargepalett trykker du update



Og fargene i “material-theme” oppdaterer seg!



Laste inn i Android Studio

Kodefilene man får eksportert fra figma er ulike fra hva Android oppretter automatisk..

Dette gjør det litt klønete å skulle legge over første gangen (som dere så sist), men her er hvordan det så ut i våre filer i prosjektet vårt i fjor

```
Resizable (Experimental) A...  app  ▶  ⚙  ⋮
M+ Readme.md  Theme.kt  Color.kt ×
1 package com.example.in2000_gruppe5.ui.theme
2
3 import androidx.compose.ui.graphics.Color
4
5 // Visual profile updated 24-04-2023
6 val Orange = Color( color: 0xFFFF934F)
7 val Midnight = Color( color: 0xFF1F313A)
8 val Dusk = Color( color: 0xFF324F5F)
9 val LightFog = Color( color: 0xFFECEEF2)
10 val OffWhite = Color( color: 0xFFFAFAFA)
11
12
13 // Default color codes
14 val Purple80 = Color( color: 0xFF08BCFF)
15 val PurpleGrey80 = Color( color: 0xFFCCC2DC)
16 val Pink80 = Color( color: 0xFFFB8C8C)
17
18 val Purple40 = Color( color: 0xFF6650A4)
19 val PurpleGrey40 = Color( color: 0xFF625B71)
20 val Pink40 = Color( color: 0xFF7D5260)
21
22 val md_theme_light_primary = Color( color: 0xFF06689)
23 val md_theme_light_onPrimary = Color( color: 0xFFFFFFFF)
24 val md_theme_light_primaryContainer = Color( color: 0xFFC3E8FF)
25 val md_theme_light_onPrimaryContainer = Color( color: 0xFF001E2C)
26 val md_theme_light_primaryFixed = Color( color: 0xFFC3E8FF)
27 val md_theme_light_onPrimaryFixed = Color( color: 0xFF001E2C)
28 val md_theme_light_primaryFixedDim = Color( color: 0xFF78D1FF)
29 val md_theme_light_onPrimaryFixedVariant = Color( color: 0xFF004D68)
30 val md_theme_light_secondary = Color( color: 0xFF06689)
31 val md_theme_light_onSecondary = Color( color: 0xFFFFFFFF)
32 val md_theme_light_secondaryContainer = Color( color: 0xFFBFE8FF)
33 val md_theme_light_onSecondaryContainer = Color( color: 0xFF001F2B)
34 val md_theme_light_secondaryFixed = Color( color: 0xFFBFE8FF)
35 val md_theme_light_onSecondaryFixed = Color( color: 0xFF001F2B)
36 val md_theme_light_secondaryFixedDim = Color( color: 0xFF6ED2FF)
37 val md_theme_light_onSecondaryFixedVariant = Color( color: 0xFF004D65)
38
39 in2000_gruppe5 > ui > theme > Color.kt
```

```
Resizable (Experimental) A...  app  ▶  ⚙  ⋮
M+ Readme.md  Theme.kt ×  Color.kt
1 package com.example.in2000_gruppe5.ui.theme
2
3 > import ...
17
18 private val LightColors = lightColorScheme(
19     primary = md_theme_light_primary,
20     onPrimary = md_theme_light_onPrimary,
21     primaryContainer = md_theme_light_primaryContainer,
22     onPrimaryContainer = md_theme_light_onPrimaryContainer,
23     // primaryFixed = md_theme_light_primaryFixed,
24     // onPrimaryFixed = md_theme_light_onPrimaryFixed,
25     // primaryFixedDim = md_theme_light_primaryFixedDim,
26     // onPrimaryFixedVariant = md_theme_light_onPrimaryFixedVariant,
27     secondary = md_theme_light_secondary,
28     onSecondary = md_theme_light_onSecondary,
29     secondaryContainer = md_theme_light_secondaryContainer,
30     onSecondaryContainer = md_theme_light_onSecondaryContainer,
31     // secondaryFixed = md_theme_light_secondaryFixed,
32     // onSecondaryFixed = md_theme_light_onSecondaryFixed,
33     // secondaryFixedDim = md_theme_light_secondaryFixedDim,
34     // onSecondaryFixedVariant = md_theme_light_onSecondaryFixedVariant,
35     tertiary = md_theme_light_tertiary,
36     onTertiary = md_theme_light_onTertiary,
37     tertiaryContainer = md_theme_light_tertiaryContainer,
38     onTertiaryContainer = md_theme_light_onTertiaryContainer,
39     // tertiaryFixed = md_theme_light_tertiaryFixed,
40     // onTertiaryFixed = md_theme_light_onTertiaryFixed,
41     // tertiaryFixedDim = md_theme_light_tertiaryFixedDim,
42     // onTertiaryFixedVariant = md_theme_light_onTertiaryFixedVariant,
43     error = md_theme_light_error,
44     errorContainer = md_theme_light_errorContainer,
45     onError = md_theme_light_onError,
46     onErrorContainer = md_theme_light_onErrorContainer,
47     background = md_theme_light_background,
48     onBackground = md_theme_light_onBackground,
49     outline = md_theme_light_outline,
50     inverseOnSurface = md_theme_light_inverseOnSurface.
51
52 in2000_gruppe5 > ui > theme > Theme.kt
```


Eksempel på hvordan man tar det i bruk

```
val costStrings = stringArrayResource(id = R.array.cost_strings)

val screenConfig = LocalConfiguration.current
val scope = rememberCoroutineScope()
val bottomSheetState =
    rememberModalBottomSheetState(initialValue = ModalBottomSheetValue.Hidden)
```

```
ModalBottomSheetLayout(
    sheetState = bottomSheetState,
    sheetContent = { AppDrawer( header: "Hva koster det å...", costStrings[0] ) },
    sheetBackgroundColor = Midnight,
    sheetShape = RoundedCornerShape( percent: 5)
) {
```

```
    Scaffold(
        topBar = {
            TopAppBar(
```

```
Column(
```

```
    modifier = Modifier.weight(4f),
```

```
) { this: ColumnScope
```

```
Text(
```

```
    text = stringResource(id = "Hva koster det å...?"),
```

```
    style = Typography.headlineMedium
```

```
)
```

```
}
```

Resizable (Experimental) A...

Readme.md Theme.kt Color.kt

```
1 package com.example.in2000_gruppe5.ui.theme
```

```
2
```

```
3 import androidx.compose.ui.graphics.Color
```

```
4
```

```
5 // Visual profile updated 24-04-2023
```

```
6 val Orange = Color( color: 0xFFFF934F)
```

```
7 val Midnight = Color( color: 0xFF1F313A)
```

```
8 val Dusk = Color( color: 0xFF324F5F)
```

```
9 val LightFog = Color( color: 0xFFECEEF2)
```

```
10 val OffWhite = Color( color: 0xFFFAFAFA)
```

```
11
```

Material 3 / Figma remote revansj - oppsummering

Det er altså ikke så farlig at det ikke er helt likt oppsett på
filene.

Dersom dere ønsker at appen skal være 100% kompatibel
med

Tips:

Datahenting / API

Hvordan henger alt sammen?

HTTP/HTTPS:

HyperText Transfer Protocol /
HyperText Transfer Protocol Secure

Protokollen som tillater
kommunikasjon mellom klienter
(f.eks. nettlesere, telefonen) og
webservere

En måte å overføre data på over
internettet

Både hente og sende data

Vi bruker vanligvis port 80

Vi bruker en HTTP-klient for å hjelpe
oss å hente data (GET)

API:

Application programming Interface

Grensesnitt for en plattform eller
tjeneste

Single Point of Entry – en måte å
aksessere på

Gjør at vi enkelt kan hente data fra
“noen andre”.

Mange APIer er åpne for alle

Vi sender en forespørsel og bryr oss
ikke om hvordan den håndterer
spørningen eller henter dataen fra
databasen - bare vi får det vi ønsker

REST:

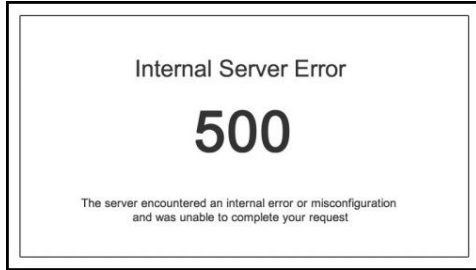
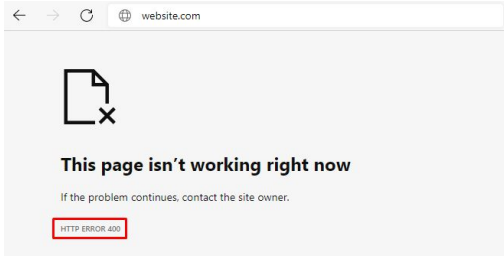
Representational State Transfer

Ikke protokoll, men arkitektur

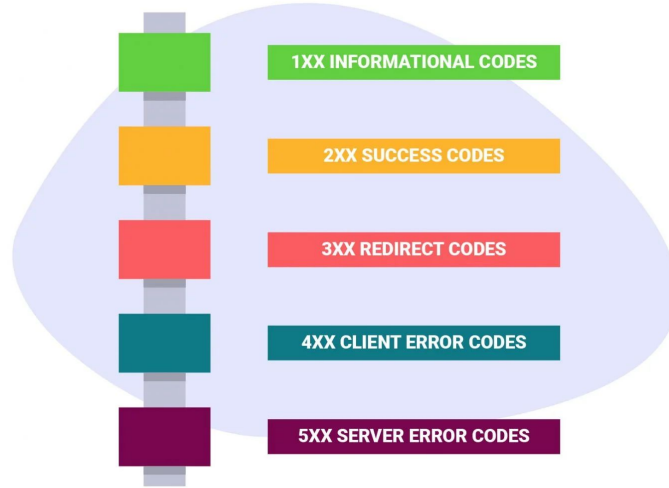
Enda en standardisert måte å hente og
sende data på, men dette bestemmer
selve grensesnittet til API-et

Et RestAPI består av ressurser som er
adressert med en URI (unik
identifikator for hver ressurs), og hver
ressurs er representert med et
dataformat (f.eks. JSON/XML)

Bruker HTTP protokollen for å få tak i
ressursen



HTTP Status Codes

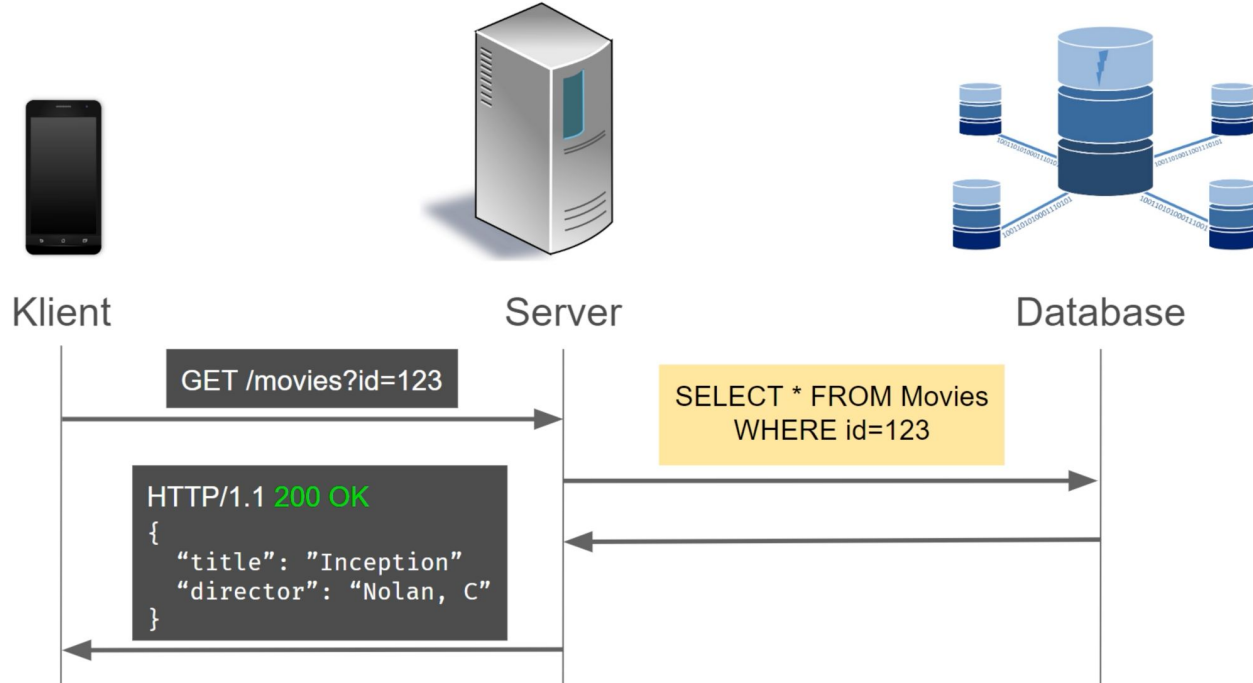


200 OK.



302 Found.

GET request



Hvordan henter vi dataen?

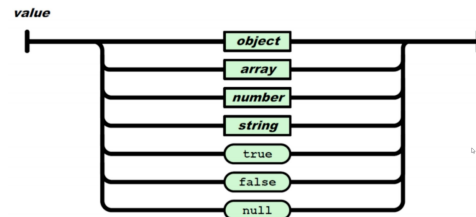
- JSON

- JavaScript Object Notation
- data i tekstformat
- en måte å strukturere og organisere data på
- samling av nøkler og verdier som er enkelt å lese og forstå
- støttes av mange forskjellige programmeringsspråk

- Postman eller Restman

- programvareverktøy som gjør det enklere å teste og utforske API-er
- kan legge inn ulike parametere og data for å se hvordan API-er

```
{
  "Students": [
    {
      "Name": "Amit Goenka",
      "Major": "Physics"
    },
    {
      "Name": "Smita Pallod",
      "Major": "Chemistry"
    },
    {
      "Name": "Rajeev Sen",
      "Major": "Mathematics"
    }
  ]
}
```



Pass på ved henting

Objekt



```
"hilsen1": "Hei",  
"hilsen2": "På",  
"hilsen3": "Deg"  
}
```

Array



```
"Hei",  
"På",  
"Deg"  
]
```

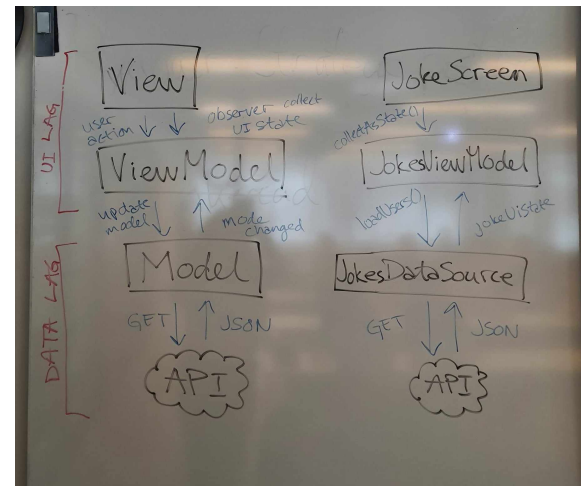
Arkitektur

MVVM - Model, View, View Model.

- Er et strukturelt designmønster for logisk oppbygging mellom Datalaget og UI-laget
- Er standard arkitektur i android apper

MVVM (og designmønstre generelt) fører til god design; bla. lav kobling og høy kohesjon(gjør det lettere å vedlikeholde, utvide og endre kode) DETTE LIKER VI!

- Vi får høy kohesjon ved å skille ulike ansvarsoppgaver i forskjellige klasser/moduler.
- Vi får lav kobling ved at klasser/moduler med ulike ansvarsoppgaver har enkle grensesnitt mellom seg. (er ikke avhengige av mange andre klasser)



View: Observerer UIstate/data i viewmodel og presenter den + Muliggjør brukerinteraksjon

ViewModel: Oppdaterer UI state hvis dataen i model endrer seg. Sier at model skal oppdatere dataen sin (ved f.eks. brukerinteraksjon fra view / eller kall på funksjon som henter ny data)

Model: Har ansvar for å hente, behandle og representere data til viewmodel.

Spørsmål om oblig 2?

Send oss en mail:
idakenene@uio.no
jamilakm@uio.no
jorgenao@uio.no
