Hvordan bruker API til å samle data

I prosjektet i IN2000 skal dere bruke bruke værdata fra `met.no`. API står for `Application Programming Interface` og er en ressurs som leverer data som kan brukes av andre applikasjoner. Dere skal bruke enten `https://api.met.no/` eller https://frost.met.no/. Et API består av flere endepunkter(kalt produkter i dokumentasjonen til met) som serverer forskjellige data. Her følger stegene du må igjennom for å kunne bruke APIet i din egen applikasjon.

1 - Finn dataen du trenger

Det første du må gjøre er å finne ut hvilken data du trenger og hvor den ligger i APIet. Under `Available Products` på `api.met.no` finner du de forskjellige ressursene du har tilgjengelig. Under dokumentasjonen for produktet finner du eksempler på en `request`. En HTTP GET request er på formen `<URL> ? key=value & key=value ...`
Eksempel: `https://api.met.no/weatherapi/oceanforecast/0.9/?lat=60.10&lon=5` betyr "hent data fra `https://api.met.no/weatherapi/oceanforecast/0.9/` hvor `lat` = 60.10 og `lon` = 5. Svaret vil være en HTTP response som inneholder data. Dataen kan være på forskjellige formater. `api.met.no` server for det mest `XML`, mens `Frost` er basert på `JSON`. Enkelte produkter kan også servere `plain text`. Før du begynner å samle og behandle data er det viktig at du har oversikt over hva APIet server og hva behovet er for din applikasjon.

2 - Hent data

Når du vet hvor du skal finne data og hva slags format(er) den er på må du hente dataen. Når du skriver inn `https://api.met.no/weatherapi/oceanforecast/0.9/?lat=60.10&lon=5` i nettleserenskjer gjør den en HTTP GET request. Dette kan gjøres programatisk fra kotlin. For å gjøre dette trenger du et HTTP bibliotek. Det er mange å velge mellom, og dere står fritt til å velge helt selv. Èn mulighet er å bruke `khttp` (kotlin-http):

```kotlin
/**
 * Simple HTTP GET using library 'khttp'
 *
 * khttp is a simple and easy to use HTTP library for Kotlin
 *
 * GitHub project: https://github.com/jkcclemens/khttp
 *
 * Gradle dependency:
 * implementation group: 'khttp', name: 'khttp', version: '0.1.0'
 */
fun main() {
    val baseUrl = "https://in2000-proxy.ifi.uio.no/yr"
    val product = "probabilityforecast/1.1/?lat=60.10&lon=9.58&msl=70"
    val response = khttp.get("$baseUrl/$product")
    println(response.text)

}
```

MERK: Når du gjør spørringer mot APIet må du bruke `https://in2000-apiproxy.ifi.uio.no/` i stedet for `https://api.met.no/`, og `https://in2000-frostproxy.ifi.uio.no/` i stedet for `https://frost.met.no/`.
Dette er for å avlaste serverne deres når vi skal ha mange teams som bruker APIet deres.

3 - Behandle data

For at det skal være mulig for applikasjonen din å bruke dataen må responsen transformeres til et Kotlin objekt.

For dette må dere modellere dataen. Generelt sett er dette vanlig objektorientert moddelering og handler for det meste om å lage klasser som representerer domenet.

Til dette skal dere bruke et bibliotek. Her kan dere også velge helt selv, men det som anbefales er `SimpleXml` for XML, og`Kotson` for JSON.
SimpleXml er et Java bibliotek som parser XML.
Se oppgavene fra Uke3 for eksempler på hvordan man bruker SimpleXml.
Kotson er en utvidelse av Gson for Kotlin. Gson er Google sitt Json bibliotek for Java.
Les: `https://github.com/SalomonBrys/Kotson`
Disse bibliotekene benytter ofte `annotasjoner` som dere legger til på klassene for at biblioteket skal kunne lage instanser av klassen utifra en streng med data.

3.5 - Retrofit

Retrofit er et større bibliotek som tar seg av både henting og transormering av data. Retrofit er standarden på Android og lar deg modellere APIet på en sammenhengende måte. Ulempen med å bruke retrofit er at du må lære et litt mer kompleks rammeverk, mens fordelen er at du slipper en del repeterende kode og får hjelp med å holde koden godt strukturert.
For mer informasjon se: `https://square.github.io/retrofit/` og oppgavene fra Uke3.

4 - Abstraher

Når du kan hente og behandle data er du klar til å lage en applikasjon. På dette stadiet burde du begynne å tenke på arkitekturen i appen din. Lage klasser som representerer klienten på en god måte slik at du enkelt kan hente daten enkelt og bruke den til fremvisning.
For mer informasjon om arkitekter i Android se ressurser fra forelesning tirsdag. 26. februar.