

1. Hvorfor har vi valgt en service – kunne det laves uden?

Når et nyt kald fra patienten er blevet sendt fra PatientApp’en, er der brug for en teknologi til at notificere personalet. Da det er prioriteret at vise konceptet fremfor performance, er der valgt en pull funktionalitet til at indlæse nye kald fra systemet. Denne pull funktionalitet kan enten laves ved at der trykkes på en knap eller lign. For så at indlæse nye kald. Men for ikke at lade personalet manuelt indlæse nye kald, kan der gøres brug af en service. Da en service kan håndtere baggrundsopgaver, vil der ikke være behov for at personalet manuelt skal interagere med app’en for at holde sig opdateret med de indkomne kald. En service i android kommer i to udgaver: En ”Started” service, hvor servicen startes af en aktivitet ved at kalde startService() hvor servicen uendeligt vil udføre én operation i baggrunden uden at returnere et resultat til kalderen. Og en ”Bound” service, hvor en aktivitet eller komponent bindes til en service, der tillader et client-server interface, hvilket gør det muligt at interagere med servicen, sende request og få resultater selv på tværs af andre processer. Og det har PatientApp’en behov, fordi PatientApp’en løbende skal hente patienternes kald fra en webserver via et WebAPI. Derfor er der valgt en bound service.

1. Hvordan fungerer den?

En service fungerer som et komponent, der kan kører i baggrunden, selv når brugeren ikke interagerer med applikationen. Ved at bruge en bound service køres servicen så længe en aktivitet er bundet til den. I figur X ses det hvordan servicen bliver oprettet i PatientApp. Til højre i diagrammet ses tidslinjen for aktiviteter og andet logik der kører i hovedtråden. Og til venstre ses tidslinjen for aktiviteter der kører i en separat tråd. Da en service som default kører i hovedtråden er der behov for at oprette en anden tråd, som servicen skal køre i, da servicen vil komme til at lave blokerende operationer. Bindingsprocessen ses i figur X hvor android frameworkets ”Service” implementeres af ”CallService” og for at tillade bindingen, skal onBind() implementeres. Denne metode returnerer en IBinder objekt, der gør det muligt for klienten at interagere med servicen. Når bindingen er sket, kaldes en metode ”showCall()” der vil lave række blokerende operationer:

1. Hente nye kald fra en webserver via et WebAPI.
2. Indlæse eksisterende kald fra lokal databasen via SQLite.
3. Gem resultaterne i servicen
4. Åben en ny dialogboks med information omkring det nye kald.

Grunden til at det er nødvendigt at indlæse eksisterende kald fra lokal databasen, er for at lave en sammenligning i forhold til om der reelt set er sket et ny kald. Resultaterne (eksisterende kald og evt. Et nyt kald) gemmes i servicen, så det kan tilgås via den aktivitet, som har bundet sig til servicen, for så herefter at vise en dialogboks tilbage på hovedtråden.

Operationerne er blokkerende fordi der sker en række intensive operationer og requests, som man som bruger af PatientApp’en ikke vil vente på – og specielt ikke af en travl sygeplejerske. Derfor skal disse operationer køres i en baggrundstråd.

Da requests til webserveren sker asynkront grundet det framework der benyttes til det, og da det kan tage noget tid at få respons tilbage, må baggrundstråden stå og vente til at operationerne er færdig, før den kan starte næste iteration. Her står den og venter hvert sekund til dette er sket, som der er illustreret i figuren ved at der tjekkes på en bool variable ”isSuccess”.

Servicen er til stor gavn for PatientApp’en og en fremragende løsning hvor det ikke kræver en brugerinterkation for at gøre brug af pull funktionaliteten. Men performance mæssigt vil en bedre løsning være brug af push funktionalitet, da man på den måde ikke forgæves brænder PatientAppen’s CPU resurser og tid på noget, hvor der ikke er blevet forespurgt om et kald.