TDS200 – Kryssplattform – Skriftlig individuell hjemmeeksamen H22

Oppgave 1 – Teoretisk kompetanse (40%)

I rollen som techlead i et selskap som driver en markedsplass for brukte retrospill, består teamet av tre fullstack-utviklere med primær kompetanse i JavaScript/TypeScript. Teamet har utviklet nettløsninger og innehar backend til den eksisterende løsningen. Mobilene våre tar en stor del av hverdagen vår og industrien for nyutviklede apper øker konstant. Dette krever mer kompetanse, særlig innenfor kryssplattform-utviklingen. Denne teoretiske delen tar sikte på å diskutere hvilken plattform/kryssplattform som skal benyttes for å gi brukerne et fullkomment tilbud av ny app hvor vi unngår økt konkurranse fra det eksisterende markedet.

Bakgrunnshistorikk for caset er at det skal utvikles en ny app. Teamet har kompetansen sin ved JavaScript/TypeScript og alle har erfaring med mobilutvikling. Det eksisterer en web-løsning med backend og web-løsningen er en plattform hvor brukere kan legge inn retrospill som de har lyst til å selge. Funksjonalitetene skal være kjøp og salg og kunne se annonsene. Utfordringen blir i det følgende å utvikle en app basert på web-løsningen, men med en effektiv og ressurssterk utvikling. Ved siden språk brukes Vue for å enklere håndtere tilstand i et program. Vue er basert på komponenter som man kan sende props til, eller som har muligheten til å håndtere sin egen state.

Framework	TTC	CPU	PreRAM	RAM	ComputedRAM	∑
Native	5	4	6	6	3	24
MAML/MD_2	4	5	5	5	4	23
NativeScript	6	6	3	3	2	20
React Native	2	1	4	4	5	16
Flutter	3	3	1	2	6	15
Ionic	1	2	2	1	1	7

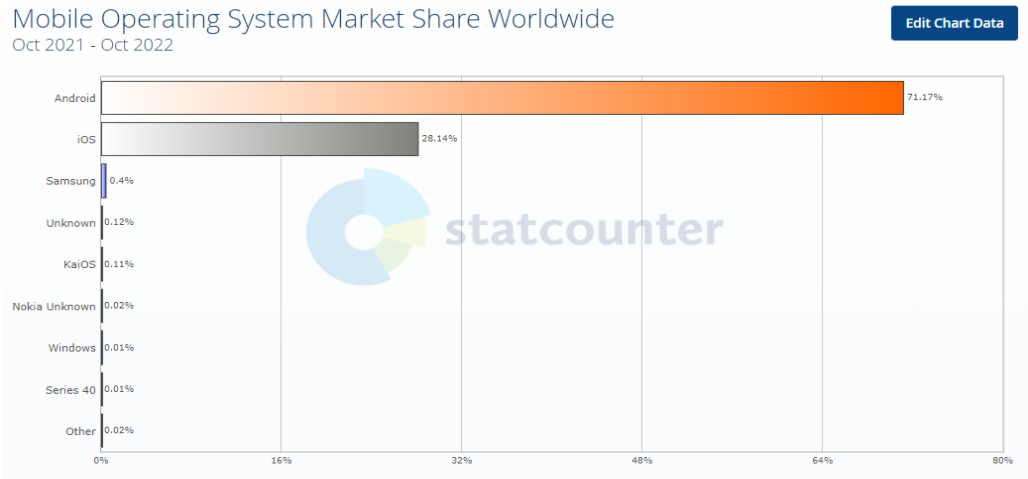
Tabell hentet fra: "*An empirical investigation of performance overhead in cross-platform mobile development frameworks*", Av: Andreas Biørn-Hansen, Christoph Rieger, Tor-Morten Grønli, Tim A. Majchrzak og Gheorghita Ghinea.

Ut ifra tabellen anvist er det flere rammeverks som er mer anbefalt enn Ionic. Gitt ut ifra spesifikasjoner og kvalifikasjoner til oppgaven er det fortsatt lønnsomt for teamet å benytte seg av Ionic uansett. Ionic benyttes fordi rammeverket tilbyr å bygge native iOS- og Android-

applikasjoner. Ionic rammeverksutvider «Capacitor» brukes ved å tilby et bibliotek for optimaliserte UI-komponenter (Nesher, 2022).

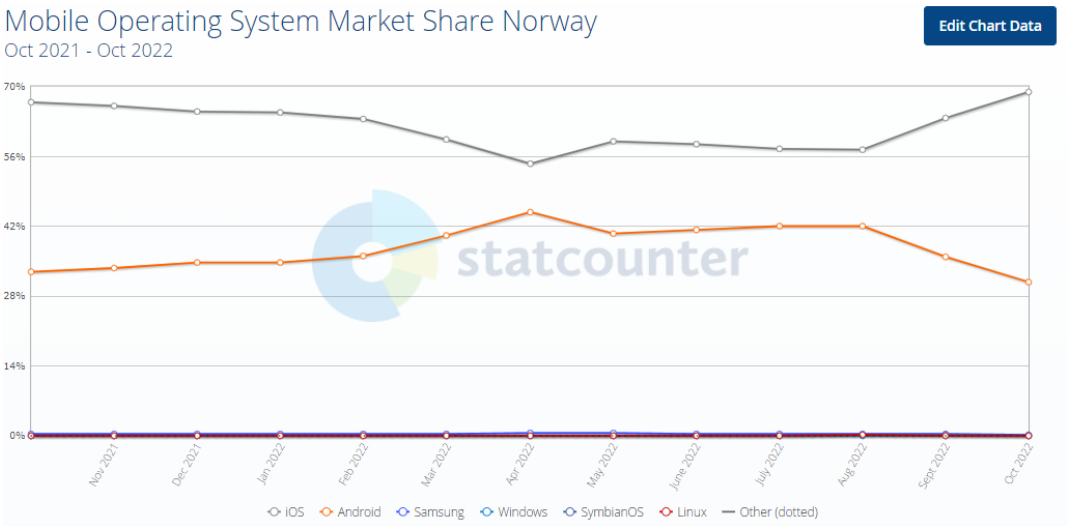
Mobilapputviklere har valget om å enten utvikle native apper for forskjellige operativsystemer uavhengig av hverandre eller utvikle å utvikle kompatible apper på tvers av plattformer. Tilgjengeligheten av forskjellige verktøy og tilnærminger for å utvikle applikasjoner for flere plattformer gjør en beslutning vanskelig. Å redusere kostnader, forenkle utvikling og opprettholde god brukervennlighet på tvers av alle operativsystemer krever kunnskap. Interessen for retrospill vokser raskt, så time-to-market er en viktig faktor når man skal utvikle en ny app. Derfor er det naturlig å se på markedsstatistikken over fordelingen av Android- og iOS-brukere i Norge.

Apple og Android sine mobiler blant de mest solgte og populære i verden, men også i Norge. På verdensbasis med tall fra 2021 av årets første kvartal viser tabellen nedenfor at Android er størst med en markedsandel på 71,2 prosent. Apple ligger et stykke bak med sine 28,14 prosent på verdensbasis.



Tabell hentet fra: *Statcounter*: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202110-202210-bar

Sett fra statistikken over hvilken leverandør som har størst utspring i Norge har Statcounter en beskrivende grad som viser forskjellene mellom Andorid og iOS. Fra perioden Oktober 2021 til Oktober 2022 vises det at markedsandelen i Norge i dag har to operativsystemer som dominerer og det er Android og iOS. Ifølge grafen ser vi at iOS (70%) er nesten 40% høyere enn Android (30%). Samtidig viser grafen at det har variert litt på populariteten for begge plattformene gjennom de siste tolv månedene. Det ville da være praktisk å kunne prioritere å forholde seg til både Android og iOS for å oppnå så mange brukere som mulig.



Tabell hentet fra: *Statcounter:* https://gs.statcounter.com/os-market-share/mobile/norway/#monthly-202110-202210

Cordova er et rammeverk for mobilutvikling kan også kan bygge HTML, CSS og JavaScript/TypeScript prosjekt til app. Dette rammeverket gir oss muligheten til å raskt gjøre om et ikke-native prosjekt til et kryssplattformprosjekt (Android+iOS), hvor all CSS og HTML-koder blir lagt til i views. JavaScript/TypeScript logikken blir videre lagt til i Java/Kotlin eller Swift/ObjC. Capacitor har en stor plass i moderne web-apper i Native og har lettere tilgang til API-er og SDK-er som ligger til grunn i disse plattformene. Denne sammenligningen tilsier at Cordova konfigurerer og bygger appen på automatikk hvor brukeren ikke trenger å gjøre noe, mens med Capacitor har man muligheten til innvirkninger. Man kan åpne de ulike prosjoketene og gjøre forandringer som for eksempel i Android Studio eller xCode for å utføre tester eller andre funksjonaliteter.

Når man skal velge hvilket rammeverk å gå for må man tenke på elementer som kravspesifikasjon, brukeropplevelse, ytelse og kildekode. Det beste utfallet er å ha en felles kodebade for Android og iOS og det gir Native oss. Native utvikling på tvers av språk gjør at det blir komplisert til native kode. Med dette grunnlaget anser selskapet det nødvendig å bruke JavaScript/TypeScript for felles kodebase. En utfordring er at native-feautres og deres tilgang er avhengig av plugins man bruker.

Selv om selskapet allerede har fastslått teknologien er det fortsatt lærerikt å sette det opp mot annen teknologi. Hybrid-apper er et alternativ hvor du lager en web-app ved hjelp av CSS, HTML og JS som kjører inne i en webview app og oppdateres deretter. De kompileres ikke ned til native, men komponentene endres slik de er kodet som. Annen kryssplattform-

teknologi kan eksempelvis være .NET for Xamarin og C#, Dart for Flutter eller JavaScript/TypeScript for Vue og React.

Hvis man ønsker å bygge app med Capacitor så vil alle kodefilene pakkes i native-appen og presenteres ved hjelp av WebView. Kommunikasjonene over broene fungerer forskjellig, rammeverkene er forskjellig også innad i «interpreted». Ulempen er at det er dyrere enn kryssplattform, krever mye tid og flere oppdateringer som kreves. Native utviklingen er en fordel da den utgir stabil ytelse, lang levetid for apper, mer skalerbarhet og fullt kompatibelt med plattformen.

Native og JavaScript/TypeScript har vært på topp i app-utviklingsverden ved å offentliggjøre muligheten for å bruke de samme UI-byggesteinene som innfødte apper. Dette gir oss en feilregistreringsfunksjon og et høyere sikkerhetsbehov (Makhov, 2022). Fordelen her blir dermed at det er færre utbyggere som kreves for utviklingen, det er ressursfattig, utviklingen tar kortere tid og muligheten for gjenbruket med kode er her. Ulempen er at det dårlig ytelse, lanseringen bruker mer tid og kodedesignet er hardere.

Hvis man ønsker å gjøre kode enda enklere enn dette og kan man benytte seg av LowCode. Benytter en seg av LowCode unngår man å bruke tid på API-implementering. Selv om dette er nytt på markedet er det mange IT-folk som ikke støtter ideen. Kompleksiteten ligger i at koden blir for lite avansert, mangel på programvareutviklingsrettigheter og skalerbarheten er ikke tilstrekkelig. LowCode er dog noe som ikke markerer en erstatter på team av fullstack-utviklere (Duque, 2021).

Valget var enkelt for dette teamet, nemlig å velge kryssplattform. Valget stod mellom å lage en kryssplattform med begge applikasjoner i en plattform gitt med med funksjonaliteter som skal støttes. Fordelen med dette valget er at det er en samlet kodebase. Alternativt kunne man hatt funksjonene i to forskjellige språk. På denne måten kan man optimalisere koden bedre. Dermed vil det være letter å endre på koden (vedlikehold) og rette bugs. Ulempen kan forekomme av problemer med kompabilitet hvor ting tar for lang tid.

#	Funksjonalitet
1	Appen må kommunisere med eksisterende backend-tjeneste.
2	Bruker av appen må kunne registrere seg og logge inn/ut (autentisering).
3	Både autentiserte brukere og ikke-autentiserte (gjester) som bruker appen må kunne se salgsannonser i listeformat.
4	Bruker av appen må kunne lese mer detaljer om annonsen ved å trykke på en annonse i listen eller på kartet (se punkt 6).
5	Innlogget/autentisert bruker av appen må kunne legge til egne annonser. Dette kan enten gjøres fra en ny side, eller fra f.eks. en popup-modal.
6	Det er ønskelig å ha en kartvisning i appen som viser lokasjonen til de forskjellige annonsene, tenk Finn.no sin kartvisning for eiendom eller torget («til salgs»).
7	Nye annonser må kunne inneholde godt med informasjon: bilde(r), tittel, beskrivelse, plattform *(PC, Nintendo, PS2, osv.)*, pris, produktets tilstand, og henteadresse (f.eks. basert på GPS-koordinater eller adresse-API).
8	Appens visuelle design skal ha sitt eget *"brandet"* preg inspirert av retrospill og retro-sjangeren.
9	Appen bør være tilgjengelig både på iOS og Android for å dekke hele mobilmarkedet.

Utviklingsopplevelsen og effektiviteten blir dårligere og da handler det bare om å få en minimal applikasjon til å virke. Selskapet må derfor inneha god arkitektur og infrastruktur.

Basert på analysen ovenfor skal selskapet benytte seg av kryssplattform-teknologi. Vue med Ionic som rammeverk og JavaScript/TypeScript som språk er en teknologi som har stor fordel ved at rammeverket tillater utviklere å lage én applikasjon (Narayan, 2021). Dette er fordi rammeverket underbygges av native API-er til plattformer og er raskere å implementere.

Kryssplattformapplikasjoner kan også regnes som tregere enn Native apper, men de egner seg godt for moderat komplekse applikasjoner. Components lar deg konstruere brukergrensesnittet for appen, samtidig som ytelse, brukeropplevelse og universell utforming overholdes i henhold til appen. Dette betyr at LowCode heller ikke er et alternativ, men at selskapet heller ønsker å benytte seg av ovenfornevnte rammeverk. På denne måten slipper utviklerne å ta ekstraordinært hensyn til nytt og komplekst design for appen som skal utvikles.

Referanser:

Duque, E. (2021, October 29). *The Good, Bad, And Ugly Of Low-Code*. Forbes. Retrieved November 17, 2022, from https://www.forbes.com/sites/servicenow/2021/10/29/the-good-bad-and-ugly-of-low-code/

Makhov, V. (2022, 06 17). *DOIT Software*. Hentet fra www.doit.software/no: https://doit.software/no/blog/app-utvikling#screen1

Narayan, A. (2021, 07 22). *CourseReport*. Hentet fra www.coursereport.com: https://www.coursereport.com/blog/react-native-vs-native-mobile-guide

Nesher, G. (2022, 01 13). *InfoQ*. Hentet fra Introducing the Ionic 6 Component Framework: https://www.infoq.com/news/2022/01/ionic-6-component-framework/

Statcounter. (2022-). *Statcounter - GlobalStats*. Hentet fra www.gs.statcounter.com: https://gs.statcounter.com/os-market-share/mobile/norway