Kryssplattformløsningsrapport

Når man velger en teknologi for kryssplattform apputvikling, er det viktig å vurdere flere aspekter, inkludert utviklingshastighet, ytelse, støtte, og tilgjengelige verktøy. Jeg fikk i oppgave å lage en Soduku-applikasjon for mobile enheter, og måtte velge en av fem mulige kryssplattformløsninger. Her følger en kort evaluering av de ulike plattformene som jeg ble presentert og hvorfor jeg valgte Flutter i mitt projekt:

1. Ionic

- **Programmeringsspråk**: TypeScript sammen med et JavaScript rammeverj. Gitt tidligere erfaringer i studiet med Vue.js, ville dette vært et naturlig valg.
- **Fordeler**:
    - **Fleksibilitet**: Mulighet til å bruke en rekke populære JS-rammeverk.
    - **Plugin-rikdom**: Et omfattende bibliotek av plugins som utvider funksjonaliteten.
- **Ulemper**:
    - **Ytelsesbegrensninger**: Selv om Ionic gir gode resultater for mange apper, kan det støte på ytelsesutfordringer, spesielt med komplekse animasjoner og UI-interaksjoner.

2. React Native

- **Programmeringsspråk**: JavaScript eller TypeScript.
- **Fordeler**:
    - **Stort Community**: Gitt populariteten til React og JavaScript, har React Native en massiv støttende community.
    - **Tilgang til native moduler**: Mulighet for å koble til og bruke native moduler når nødvendig.
- **Ulemper**:
    - **Integrasjonsutfordringer**: Integrering av native komponenter kan noen ganger være utfordrende og tidkrevende.

3. Flutter

- **Programmeringsspråk**: Dart
- **Fordeler**:
    - **Høy ytelse**: Ved å kompilere direkte til maskinkode gir Flutter en nesten-native ytelse, noe som er kritisk for applikasjoner som krever sømløse animasjoner og interaksjoner.
    - **Konsistent UI**: Med et rikt bibliotek av widgets kan utviklere enkelt skape konsistente brukeropplevelser på tvers av enheter og plattformer. Dette gir et uniformt utseende uavhengig av om brukeren bruker f.eks. IOS eller Android.
    - **Rask Utvikling**: "Hot reload" lar utviklere se endringer umiddelbart, noe som dramatisk forbedrer utviklingstiden.
    - **Enkel integrasjon med native kode**: Til tross for at Flutter bruker Dart, tillater det enkel integrasjon med native kode, noe som gir ytterligere fleksibilitet.
- **Ulemper**:
    - **Dart**: Selv om Dart er et robust og effektivt språk, er det ikke så utbredt som andre språk. Dette kan påvirke tilgjengelige ressurser og støtte. Men språket har hatt en stor økning i popularitet i nyere tid.

4. Cordova

- **Programmeringsspråk**: HTML, CSS og JavaScript.
- **Fordeler**:
    - **Gjenbruk**: Mulighet for å gjenbruke eksisterende webkunnskap og kode.
- **Ulemper**:
    - **Ytelsesbegrensninger**: Mens Cordova gjør det mulig å lage apps raskt ved å bruke webteknologier, kommer det med ytelseskostnader, spesielt sammenlignet med native apps.

5. Kirigami

- **Programmeringsspråk**: QML og C++.
- **Fordeler**:
  - **Fleksibilitet**: God tilpasningsevne for både desktop og mobile plattformer.
- **Ulemper**:
  - **Mindre Community**: I forhold til de andre alternativene har Kirigami et mindre samfunn, som kan begrense støtte og ressurser.

Basert på informasjonen over og en nøye vurdering av de ulike kryssplattformløsningene og deres egenskaper, valgte jeg Flutter for mitt prosjekt. Årsakene til dette er:

- **Ytelse**: Flutter kompilerer direkte til maskinkode, som gjør den raskere enn noen av de andre valgene. Dette gjør applikasjoner mer scalable og øker brukeropplevelse, selv med tunge operasjoner.
- **UI-konsistens**: Med Flutters omfattende bibliotek av widgets kan jeg enkelt lage en konsistent og moderne brukeropplevelse på tvers av ulike plattformer. Dette sikrer at appen føles lik, uavhengig av om den kjøres på Android eller iOS.
- **Rapid Development**: "Hot reload" gjør utviklingsprosessen lettere ettersom det gir meg muligheten til å se endringer i sanntid.
- **Språk**: Selv om Dart ikke er så populært som JavaScript, har jeg funnet at det gir flere fordeler. Dets sterke typesystem kan redusere sjansen for feil, forbedre kodekvaliteten og gjøre koden mer lesbar. Dette gir en mer robust applikasjon og var en gyllen mulighet til å lære meg et nytt språk.
- **Integrering med native kode**: Dart gir meg fleksibiliteten til å enkelt integrere med native kode når det er nødvendig, og dermed utnytte plattformspesifikke funksjoner eller biblioteker.

React Native var også et sterkt alternativ gitt sin popularitet, bruk av JavaScript og store community. Til tross for disse styrkene, følte jeg at Flutters direkte kompilering til maskinkode, dens konsistente UI-kapabiliteter, og styrken til Dart som et språk, ga det en fordel, spesielt for mitt prosjekts behov.