**Engineer** Programvare

# Bygde en native macOS-klient i Swift 6

Bygde en native meldingsklient for macOS i Swift 6 og SwiftUI — 11 141 linjer fordelt på 53 filer — over C-grensesnittet til en Rust-kjerne lenket inn som et statisk arkiv fra en låst revisjon.

august 2026

**11,141**

linjer Swift over en Rust-kjerne

**Situasjon.** Meldingsprotokollen hadde en godt testet kjerne skrevet i Rust og klienter på flere plattformer, men skrivebordsopplevelsen på macOS var et plattformuavhengig skall — det så ikke ut som en Mac-app, oppførte seg ikke som en, og hadde med seg et kjøretidsmiljø som en native app ikke trenger. Kjernen eksponerer funksjonaliteten sin gjennom et C-grensesnitt, og det betyr at ethvert språk som kan kalle C, kan bygge på den, og det kan Swift.

**Oppgave.** En native klient måtte bygges direkte på dette C-grensesnittet, i plattformens gjeldende språk og gjeldende rammeverk for brukergrensesnitt, med kjernen lenket inn i stedet for levert ved siden av.

**Handling.** Appen er 11 141 linjer Swift fordelt på femtitre filer i app-målet, bygd på SwiftUI med kjernen lenket inn som et statisk arkiv kompilert fra en fastlåst oppstrømsrevisjon. Å låse revisjonen i stedet for å følge en gren er beslutningen som gjør byggingen reproduserbar: C-grensesnittet er kontrakten, og en kjerne i bevegelse endrer kontrakten uten at én linje Swift endres. Lagdelingen holder den usikre flaten liten — en tynn wrapper eier hver peker og hver streng som krysser grensen, modellene over den er vanlige Swift-verdier, og grensesnittlaget ser aldri en rå peker. Alt C-grensesnittet returnerer, har en eierskapsregel, og å ta feil av én gir enten en lekkasje eller en krasj uten noen kompilatoradvarsel imellom, så wrapperen er der hele prosjektets minnedisiplin bor. Byggingen drives av en oppgavekjører med femtiseks mål som dekker kompileringen av kjernen, Swift-byggingen, testsuitene, linterne og pakkingen.

**Resultat.** En native klient som starter som en Mac-app, bruker plattformens egne materialer og kontroller og ikke har med seg noen innebygd nettleser. Det som ikke er gjort, bør sies: dette er ikke en levert utgivelse. Det finnes ingen notarisert distribusjon, ingen oppdateringskanal og ett språk i grensesnittet, så det er en fungerende app og ikke et produkt noen andre kjører.

### Bruk erfaringen i prosjektet deres

Bruk «Bygde en native macOS-klient i Swift 6» som utgangspunkt for en lignende utfordring i organisasjonen deres. Tjenestene bak arbeidet er tilgjengelige for prosjektet deres. Formen på et system tegnet opp på papiret først, slik at delene passer sammen og kostnaden er kjent før noen skriver kode.

**Se tjenestene våre** Ta kontakt →

### Se bak resultatet

Les om situasjonen, beslutningene og resultatet i eksemplet. Vi kan drøfte hva som passer deres rammer, og hva som krever en annen tilnærming. Bygde en native macOS-klient i Swift 6 — 11,141 linjer Swift over en Rust-kjerne

**Se tjenestene våre** Ta kontakt →

## Trenger du en ekte native app, ikke en innpakket nettside?

Snakk om prosjektet med ingeniøren som har ansvaret for leveransen. Vi avtaler omfang, prisgrunnlag og overlevering skriftlig før arbeidet begynner.

Rolle i oppdraget: Applikasjonsingeniør for macOS. Det bygger på Løsningsarkitektur, Frontend-utvikling, Fullstack-utvikling, API-er og integrasjon, Designsystemer og brukergrensesnitt og UX/UI-design. Og nå som dere er her: vi er ganske gode på Plattform- og løsningsarkitektur, Frontend-utvikling, Fullstack-produktutvikling og UI/UX-design og designsystemer.

**Ta kontakt** Se tjenestene våre

## Ett stykke av **en lengre oversikt** — og den ligger i sin helhet her.

Hver oppføring er skrevet på samme måte: situasjonen, oppgaven, hva vi gjorde, og hva som endret seg.

**Se hele oversikten**

Definer forbedringen dere trenger

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

**Se hva det koster** Ta kontakt →

## Trenger du en ekte native app, ikke en innpakket nettside?

Spør om beslutningene bak «Bygde en native macOS-klient i Swift 6». Dere snakker med folk som kan forklare arbeidet og vurdere deres eget prosjekt.

**Se tjenestene våre**

welcome@engineer.company +45 50 10 61 84 København, Danmark

Ta med deg denne siden

Visittkort

**Laget med kjærlighet av D.**

**CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N**

**Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.**

**Skrevet ut fra https://software.engineer.company/nb/portfolio/built-a-native-macos-client-in-swift-6/**