Underbilag 3.1
Visningsklienterne

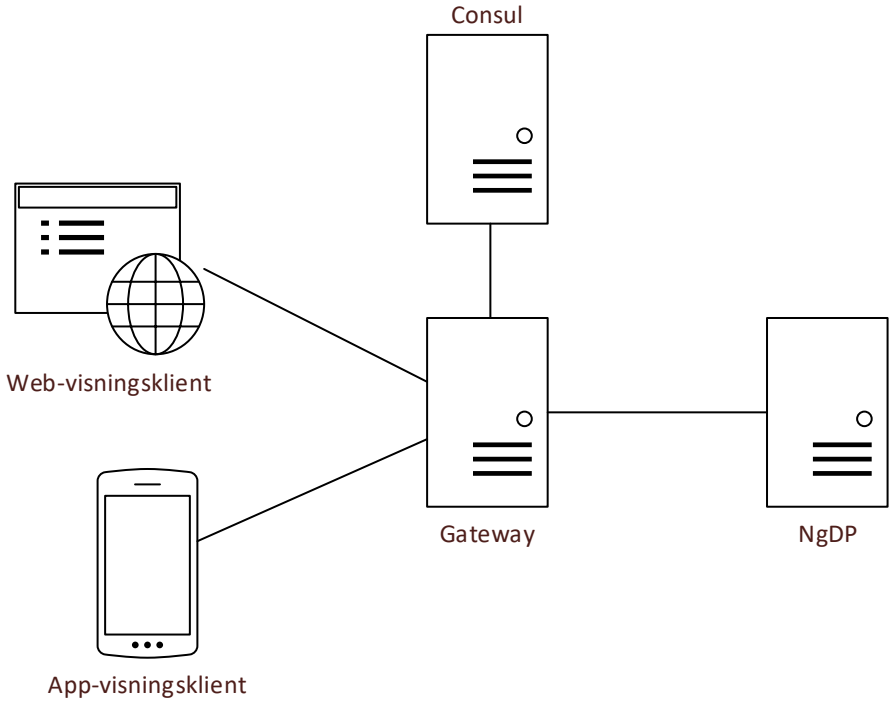
Borger.dk udbud
Maj 2021

Vejledning
Underbilag 3.1 indeholder en beskrivelse af Kundens eksisterende visningsklienter.

Indhold

1. Indledning

Borger.dk har to visningsklienter til Digital Post, som begge integrerer op mod Næste generation Digital Post (NgDP). Visningsklienterne består af en web-visningsklient, en app-visningsklient, og en API gateway, som begge visningsklienter benytter til kommunikation med NgDP.



2. App-visningsklient

2.1 Overordnet beskrivelse af softwareløsningerne

Udviklingen af app-visningsklienten blev påbegyndt i slutningen af 2020 og er fortsat under udvikling med release pt. planlagt i efteråret 2021. Appen er en visningsklient for Digital Post og henter således al borgerens data fra Næste Generation Digital Post systemet (NgDP). Den udvikles til brug på både Android og iOS med understøttelse af telefon, tablet og diverse tilgængelighedsindstillinger såsom skalérbart indhold og skærmlæser.

Appen udgives i hhv. Apple AppStore og Google Play Store.

2.1.1 Rolle og komponenter

App-visningsklientens rolle er:

1. Visning af meddelelser fra NgDP, herunder:
    a. Postkasser tilknyttet brugeren inklusiv disses beskeder og mapper
    b. Beskeder fra myndigheder inklusiv eventuelt forkyndelse heraf
2. At udstille funktionalitet i MeMo formatet, herunder MeMo-handlinger mv.
3. At være en platform til at foretage handlinger vedrørende Digital Post, f.eks.:
    a. Foretage handlinger på beskeder fra myndigheder
    b. Sende nye/besvare/videresende beskeder
    c. Se/downloade/printe tilføjede filer
    d. Modtage push beskeder og vise relevant indhold i appen på baggrund af definerede events i Digital Post

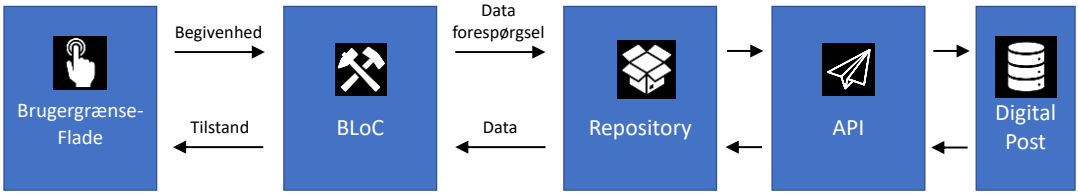
Appen understøtter ikke brugere, der ikke allerede har en digital postkasse og linker her i stedet til web-visningsklienten i forbindelse med support for tilmelding.

Appen er en tynd klient og gemmer derfor ikke egentlig data på brugerenes enheder. Som udgangspunkt gemmes kun dét, der er nødvendigt for at supportere hurtigt log-in for tilbagevendende brugere. I stedet integrerer appen med Digital Post via en Gateway (se afsnit 4), hvorfra data udstilles og hentes, når appen kræver det.

Appen udvikles som en cross-platform app vha. teknologien og platformen Flutter. Det betyder i praksis, at det er samme kodebase for hhv. android og iOS. Appen afhænger af en række Flutter plugins, der understøtter både appens arkitektur samt funktionelle krav.

2.1.2 Arkitektur

Flutter betragter overordnet UI'et som en funktion af den gældende tilstand (function of state), hvilket i praksis betyder, at UI'et er datadrevet, og ikke som sådan manipuleres eller ændres direkte. For at supportere dette er appens arkitektur baseret på BLoC (Business Logic Component) design, hvor forretningslogik separeres fra brugergrænsefladens komponenter gennem såkaldte BLoCs, der håndterer alle begivenheder startet af brugeren samt appens resulterende tilstande.



I arkitekturen starter realiseringen af en handling (f.eks. at slette en besked) typisk med, at en begivenhed registreres fra brugergrænsefladen (f.eks. at brugeren trykker på knappen "Slet besked"). Begivenheden registreres i den ansvarlige BLoC, der udfører den tilhørende logik. Hvis logikken involverer håndtering af data (f.eks. at slette en besked), sker dette i data repositories, der via dedikerede RESTful API klienter kan kalde relevante endpoints hos Digital Post. Som følge af resultatet af logikken, udsteder den ansvarlige BLoC en ny tilstand, som brugergrænsefladen derefter kan reagere på.

Udover brugerens data hentes også de i løsningen anvendte datamodeller, konfigurationer og oversættelser direkte fra Digital Post systemet. Implementeringen af gateway endpoints og deres datamodeller bygger primært på den udstillede swagger definition hvorfra app-modellerne autogenereres.

Appens UI er lavet sådan, at det tilpasser sig skærmstørrelse og evt. tilgængelighedsindstillinger i form af f.eks. større skriftstørrelse. Herudfra justeres de grafiske elementer, sådan at ratio mellem elementerne bevares ved visning på skærmen.

Autentificering af brugeren foregår med OpenID Connect, der tillader autorisationsserveren hos Digital Post at identificere brugeren via standardprotokollen OAuth 2.0 pt. med NemLog-in som identity provider. Brugerdata vedrørende autentificering og login gemmes lokalt på appens enhed – f.eks. autorisationstoken og refresh token i forbindelse med OIDC, hvor sikkerheden er håndteret ved brug af Keychain til iOS og AES kryptering til Android.

Appen logger pt. ikke fejl eller problemer. Dog pågår der undersøgelse af hvorvidt det vil være en mulighed at benytte cloudplatformen Sentry.io til overvågning og rapportering af fejl.

Brugergrænsefladen er implementeret ud fra det af DIGST designmæssige udførte design visualiseret vha. Figma – med fokus på genanvendelige designkomponenter ud fra et specificeret designsystem.

2.2 Overblik over Softwareløsningen

App-visningsklienten forventes ved projektets afslutning at bestå af følgende dokumentation:

- D0160 – User-Interface Design
- D0180 – External Interface Design
- DD120 – Physical Datamodel
- DD130 – Detailed Design
- DD160 – Programming Guidelines
- O0210 – Data Security Plan
- O0300 – Maintenance Guide
- O0500 – Software Architecture

2.2.2 Overblik over udvalgte plugins

Appen afhænger af en række Flutter plugins, der både understøtter arkitekturen og funktionelle krav. Følgende tabel beskriver udvalgte plugins, der understøtter løsningens arkitektur:

Plugin	Beskrivelse
flutter_secure_storage	Sikker opbevaring af data på enhed. Bruger Keychain til iOS og AES kryptering til Android.
Dio	HTTP klient til understøttelse af en RESTful API.
flutter_bloc	Alle komponenter til BLoC arkitekturen er indeholdt i dette plugin.
get_it	Dependency injection (af eksempelvis storage komponent).
flutter_appauth	Implementering af OpenID Connect med hjælpemetoder til at kalde autorisationsserveren hos Digital Post.
flutter_screenutil	Skalering af indhold baseret på skærmstørrelse.
file_picker + image_picker	Native understøttelse af valg af hhv. filer og billeder

2.3 Teknologier

I det følgende sammenfattes de væsentligste teknologier og komponenter, som løsningen benytter:

- Flutter 1.22.6 (software development kit bygget på udviklingssproget Dart)
- Dart 2.10.5 (udviklingssprog)
- RESTful API (for integration med Digital Post)
- OpenID Connect (autentificering via OAuth 2.0)
- Sentry.io (fejlrapportering) – undersøgelse pågår
- Azure DevOps Server (kodebase og understøttelse af agile processer)

3. Web-visningsklient

3.1 Overordnet beskrivelse af systemet

Web-visningsklienten ligger under borger.dk rammen, den har været under udvikling siden slutningen af 2019 og forventes færdigleveret ved udgangen af 2021. Web-visningsklient projektet har til opgave at levere en moderne og opdateret webklient til visning af Digital Post i den offentlige sektor.

3.1.1 Rolle og komponenter

Web-visningsklientens primære roller er:

- Visning af meddelelser fra Næste Generation Digital Post (NgDP)
- At udstille funktionalitet i MeMo formatet, herunder Memo handlinger, forkyndelser mv.
- At udstille funktionalitet hos NgDP (adviseringer, hændelseslog mv.)

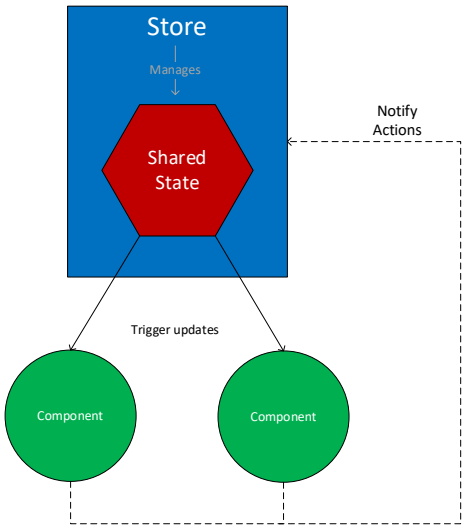
For at understøtte ovenstående er der blevet bygget en frontend applikation i Vue, som interagerer med NgDP gennem en Gateway (se afsnit 3)

Når en bruger logger ind, hentes en række oplysninger omkring brugeren og dennes postkasse, som er nødvendigt til visning af data i webklienten.

3.1.2 Arkitektur

I webklienten benyttes FLUX arkitekturen, hvor applikationens overordnede state er styret i VUEX og alle actionflows er ensrettet, hvilket giver et simplere

workflow i applikationen. Overordnet følger applikationen logikken, at en Action, kan trigger en Mutation, som opdaterer applikationens State. Komponenter kan subscribe til ændringer i applikationens state og foretage relevante opdateringer på baggrund af disse ændringer.



Klientens konfiguration er defineret i Consul (Se afsnit 3) og bliver hentet ind ved load af klienten, det samme gør sig gældende for alle tekster i klienten, der ikke er kommer fra NgDP. Ved load af klienten hentes en liste af endepunkter ligeledes fra Consul og gemmes i VUEX storen.

3.2 Overblik over softwareløsningerne

Klienten består af en række forskellige Softwareløsninger, som tilsammen udgør Systemet.

I det følgende findes en beskrivelse af Kundens Softwareløsninger:

- D0160 – Brugergrænseflade design
- D0130 – Detaljeret design
- O0400 – Teknisk infrastruktur
- O0300 – Vedligeholdelsesvejledning

3.2.1 Overblik eksterne biblioteker

Webklienten benytter en række eksterne biblioteker til understøttelse af løsnings funktionelle krav:

Plugin	Beskrivelse
Vue.js	Web framework
Vuex	State management

Vue Router	Rutehåndtering
Bootstrap	CSS-framework
Jest	Unittesting
PDF.js	Visning af PDF-vedhæftninger i visningsklienten

3.3 Teknologier

I det følgende sammenfattes de væsentligste teknologier, som Systemet benytter:

- HTML/CSS
- JavaScript
- Vue
- Restful Api
- OpenID Connect
- Azure Devops (Repo og opgavestyring)

4. Gateway

4.1 Overordnet beskrivelse af systemet

4.1.1 Rolle og komponenter

Gatewayens primære roller er:

- Simplificering og aggregering af kald fra visningsklienterne til NgDP
- Logning af fejl
- Udstilling af konfigurationer og faste tekster til visningsklienterne
- Understøttelse af BULK funktionalitet i visningsklienterne
- OIDC-klient framework for webklienten
- OIDC-proxy for App visningsklienten

4.1.2 Arkitektur

Løsningen er bygget op omkring Ocelot, som er et API gateway produkt bygget til .Net. Ocelot fungerer ved, at den tager et request ind og matcher det mod en liste af konfigurerede endpoints. Hvis den finder et relevant endpoint (upstream), findes der i samme konfiguration et relevant endpoint, som den skal kalde (downstream). Efter en request er matchet mod listen af upstream endpoints, køres requesten gennem en stak af forskellige handlers, som kan modificere requestet, inden det sendes videre til downstream endpointet. Når der kommer

svar tilbage, bliver requesten igen kørt gennem en stak af handlers, som igen modificerer svaret, inden det sendes tilbage til visningsklienterne.

Ocelot gør det også muligt at lave BULK handlinger i visningsklienterne, således at klienterne kan sende et enkelt kald, som resulterer i flere kald ud fra gatewayen. Gatewayen vil dernæst vente på svar på alle downstream kald, inden den sender et aggregeret svar tilbage til visningsklienterne.

Consul benyttes til at opbevare konfigurationer, faste tekster, Api endpoints mv. som visningsklienterne kunne have behov for. Derudover indeholder Consul konfigurationer til API gatewayen, som kan ændres runtime uden behov for deploy og nedetid.

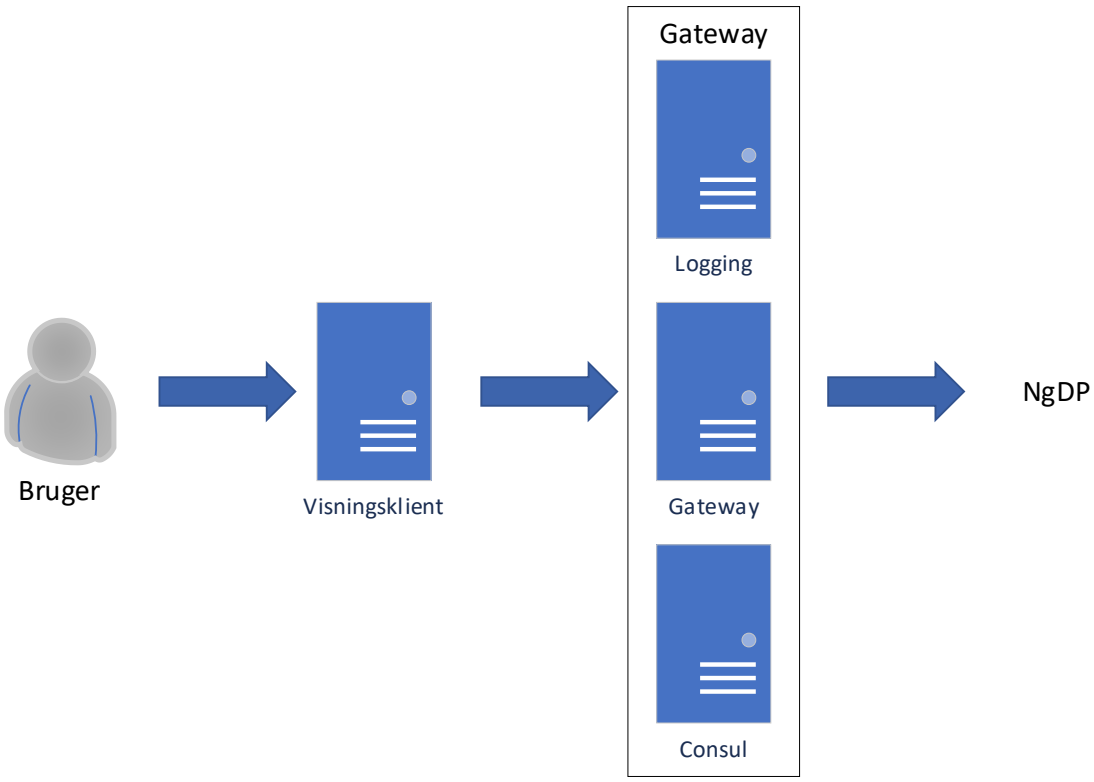
4.2 Overblik over softwareløsningerne

Gatewayen består af en række forskellige Softwareløsninger, som tilsammen udgør Systemet.

I det følgende findes en beskrivelse af Kundens Softwareløsninger:

- D0130 – Detaljeret design – sikkerhed
- D0130 – Detaljeret design
- O0400 – Teknisk infrastruktur
- O0300 – Vedligeholdelsesvejledning

4.2.1 Overblik over komponenter



4.2.2 Beskrivelse af de enkelte komponenter

Visningsklient

Denne kan både være webklienten og app, begge sender requests til NgDP gennem gateway, og begge benytter Consul til konfigurationsstyring og lagring af faste tekster.

Gateway

Gatewayen består af en .net core applikation der benytter Ocelot til requesthåndtering, dens endpoint konfiguration ligger gemt i Consul. Konfigurationen af Ocelot sker i Consul hvilket gør det muligt at opdatere håndtere ændringer hos eksterne integrationer uden behov for deploy.

Consul

Consul indeholder klientkonfigurationer, faste tekster og rutekofigurationer. Consul giver mulighed for opdatering af klienternes indstillinger og tekster i runtime uden behov for deploy. Consul indeholder også mapping og konfiguration af snitfladerne, hvilket gør det muligt at styre systemets opførsel (Curcuitbreaking, throtteling og fejlhåndtering) runtime uden behov for at deployere gatewayen.

Logning

Systemets centrale overvågning er baseret på et standardprodukt (ELK-stak)

4.3 Teknologier

I det følgende sammenfattes de væsentligste teknologier, som Systemet benytter:

- Ocelot
- Consul
- .Net Core
- ElasticSearch
- Kibana
- Logstash

digst.dk