

Projektrapport

BridgeControl



Dato: 27-09-2026

Skole: TEC

Titelblad

Deltagere

Vejleder:

Thomas Dam Andersen

Deltagere:

Amein Saidi - amein.saidi@gmail.com

Amein Saidi

Jonas Offersen - zelanard@live.co.uk

Jonas Offersen

Indholdsfortegnelse

Titelblad	2
Indholdsfortegnelse	3
Læsevejledning	5
Indledning	7
Hovedkonceptet	8
Softwaredistribution	8
Konfigurationsstyring	9
Konfigurationsmigration	9
Brugere og adgang	10
Overstyring og overvågning	10
Typisk workflow	10
Problemformulering	12
Kravspecifikation	13
Funktionelle krav og User Stories	13
Milestone 1 – Foundation	13
Milestone 2 – Software Catalog	13
Milestone 3 – Agent Connectivity	13
Milestone 4 – Configuration Management	13
Milestone 5 – Configuration Migration	14
Milestone 6 – Software Deployment	14
Milestone 7 – Monitoring	14
Milestone 8 – Audit & Hardening	14
Brugervejledning	14
Agent	14
Produkt	14
Licens	15
Konfiguration og installation	15
Overvågningsoversigten / Installationernes tilstand	15
Connectivity – forbindelsen til installationens Agent	15
Freshness – aktualiteten af installationens seneste rapport	15
Configuration – klargøring af installationens effektive konfiguration	16
License – installationens licensberettigelse	16
Deployment – installationens seneste deployment	16
Teknisk produktdokumentation	16
Formål og systemarkitektur	16
Software og teknisk driftsmiljø	17
Datamodeller og adskillelse af leverandører	17
HTTP API og adgangskontrol	18

Dato: 27-09-2026

Skole: TEC

Konfiguration og versionsstyring	18
Udrulning og Agentens arbejde	19
Overvågning og belastningsberegning	19
Testrapport	20
Testmetoder og testresultater	20
Projektplanlægning	21
1. Formål	21
2. Udviklingsmetode	21
3. Product Backlog og milestones	22
4. User Stories	22
5. Foreløbig tidsplan	22
6. Opgavestyring	23
7. Løbende evaluering og ændringer	23
Arbejdsfordeling	24
Sprint 0	24
Uge 3 og 4	24
Uge 5	24
Begrundelse for metodevalg og teknologi	24
Agil udviklingsmetode	24
Backend	25
Frontend	25
Projektdagbog/logbog	25
Realiseret tidsplan	26
24.-28. august – Projektstart og projektafgrænsning	26
31. august-7. september – Kravspecifikation og projektplanlægning	26
8.-11. september – Systemdesign og teknisk planlægning	26
14.-20. september – Udvikling af backend, Agent og frontend	27
21.-23. september – Færdiggørelse af frontend og integration	27
24.-25. september – Rapportskrivning og afslutning	27
Afgivelser fra den foreløbige tidsplan	27
Konklusion	28
Refleksion	29

Læsevejledning

Denne rapport dokumenterer udviklingen af BridgeControl og beskriver både det udviklede system og det projektforløb, der ligger bag løsningen. Rapporten er opbygget, så læseren først introduceres til projektets formål og BridgeControls grundlæggende funktionalitet, hvorefter rapporten gradvist går mere i dybden med krav, anvendelse, teknisk implementering, test og projektforløb.

Rapporten indledes med en introduktion til BridgeControl og et overblik over systemets centrale funktioner. Her beskrives blandt andet hovedkonceptet, softwaredistribution, konfigurationsstyring og -migration, brugeradgang, overvågning samt det typiske workflow i systemet. Formålet med denne del er at give læseren en grundlæggende forståelse af BridgeControl, inden de mere detaljerede dele af rapporten gennemgås.

Herefter præsenteres projektets problemformulering og de tilhørende underspørgsmål. Problemformuleringen danner udgangspunkt for projektet og fokuserer blandt andet på administration af flere softwareforhandlere, konfiguration, automatiseret installation og opdatering, overvågning samt sikker kommunikation. Kravspecifikationen konkretiserer efterfølgende systemets funktionelle behov gennem otte milestones og de tilhørende User Stories.

I brugervejledningen beskrives den praktiske anvendelse af systemet. Her gennemgås blandt andet opsætning af Agenten, oprettelse af produkter og licenser, konfiguration og installation samt anvendelsen af overvågningsoversigten. Denne del kan derfor læses som en praktisk introduktion til de vigtigste arbejdsgange i den udviklede løsning.

Den tekniske produktokumentation går derefter i dybden med løsningens tekniske opbygning og implementering. Her beskrives systemarkitekturen, det tekniske driftsmiljø, datamodeller og adskillelse af leverandører, HTTP API'et og adgangskontrol, konfigurations- og versionsstyring, udrulning, Agentens arbejde samt overvågning og belastningsberegning. De tekniske beskrivelser suppleres af diagrammer og API-dokumentation i bilagene.

Efter den tekniske dokumentation følger testrapporten, som beskriver de automatiserede og manuelle testmetoder, der er anvendt til at kontrollere systemets funktionalitet. Her behandles blandt andet unit tests, integrationstests og manuel test samt begrænsningerne ved de forskellige testformer.

Rapportens efterfølgende del fokuserer på selve projektforløbet. Projektplanlægningen beskriver den valgte agile arbejdsmetode, Product Backlog, User Stories, den foreløbige tidsplan og den løbende opgavestyring. Arbejdsfordelingen beskriver, hvordan udviklingsopgaverne blev fordelt mellem gruppens medlemmer, mens afsnittet om metode- og teknologivalg begrundet de centrale valg, der blev truffet under udviklingen.

Projektdagbogen og den realiserede tidsplan bruges til at dokumentere projektets faktiske forløb og til at sammenholde det med den oprindelige planlægning. Her beskrives blandt andet projektets udviklingsfaser, arbejdsfordeling, tekniske udfordringer og de ændringer, der opstod undervejs.

Rapporten afsluttes med en konklusion, hvor projektets resultater sammenholdes med problemformuleringen, samt en refleksion over løsningens begrænsninger og muligheder for videreudvikling. Refleksionen omfatter blandt andet understøttelse af flere operativsystemer,

Dato: 27-09-2026

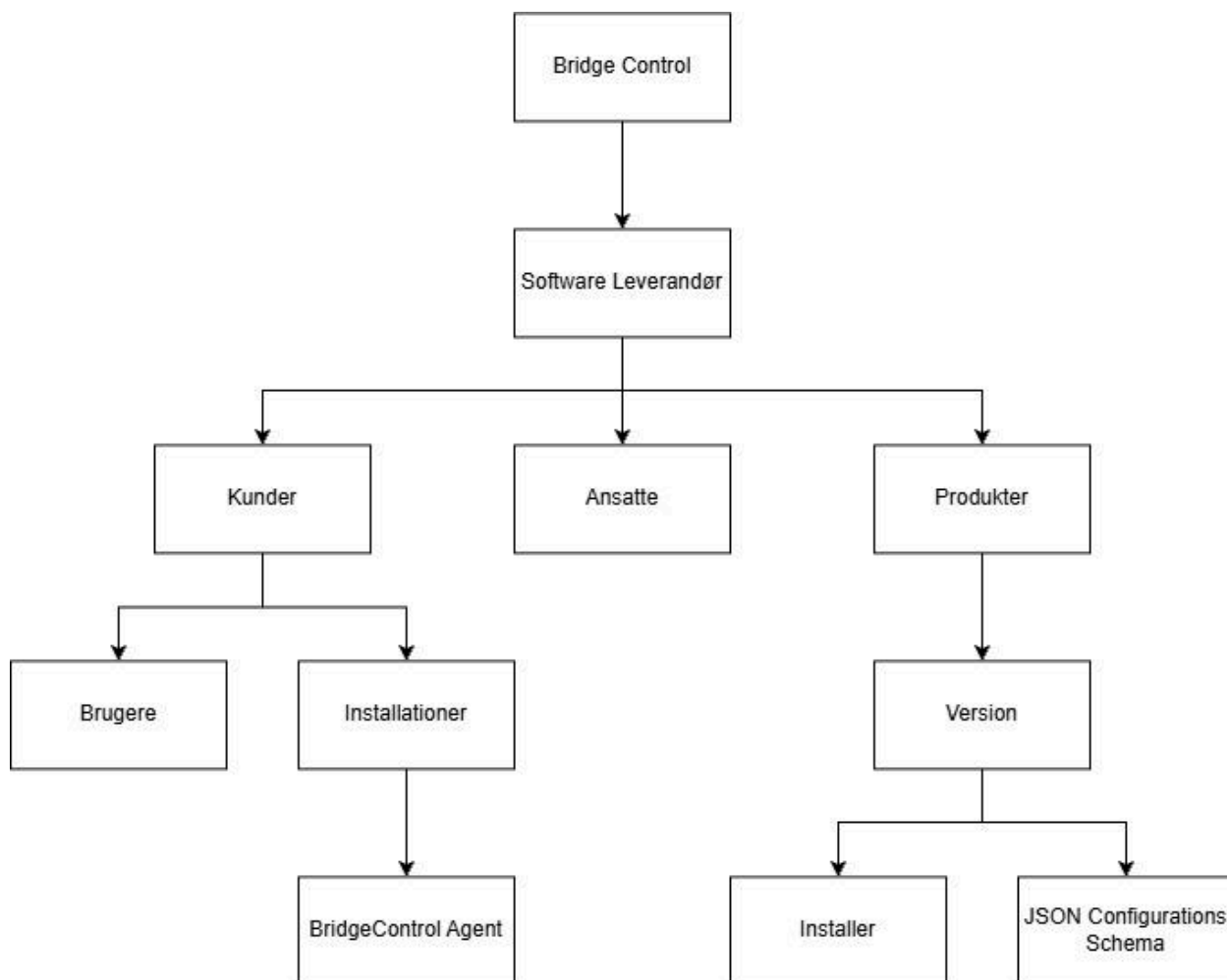
Skole: TEC

installations- og konfigurationsformater, automatisk opdatering af Agenten samt forbedringer af brugergrænsefladen.

Rapporten skal læses i sammenhæng med de tilhørende bilag. Bilagene indeholder supplerende materiale, som ville være for omfattende at medtage direkte i rapporten, herunder User Stories, Product Backlog, projektdagbog, tekniske diagrammer og HTTP API-dokumentation. Der henvises løbende til disse bilag, hvor de understøtter eller uddyber rapportens indhold.

Indledning

BridgeControl er en central software management platform, der gør det muligt for softwareleverandører, både enkeltstående udviklere og virksomheder, at distribuere software til flere kunder. Systemet giver en leverandør mulighed for at holde styr på sine produkter, kunder, installationer, konfigurationer, licenser og opdateringer fra ét sted.



Dato: 27-09-2026

Skole: TEC

Hovedkonceptet

BridgeControl adskiller softwarens ønskede tilstand fra dens faktiske tilstand.

Før hver installation på platformen:

- Hvilket produkt der skal installeres.
- Hvilken version installationen skal køre.
- Hvordan produktet skal konfigureres.
- Om produktet har en gyldig licens.

Agenten på kundens maskine rapporterer:

- Den nuværende installationsversion.
- De aktive konfigurationer.
- Om installationen er online.
- Om installationer og opdateringer er lykkedes.
- Relevante fejl og logfiler.

Dette gør det muligt for BridgeControl at identificere installationernes aktuelle tilstand og bringe dem til den ønskede tilstand.

Softwaredistribution

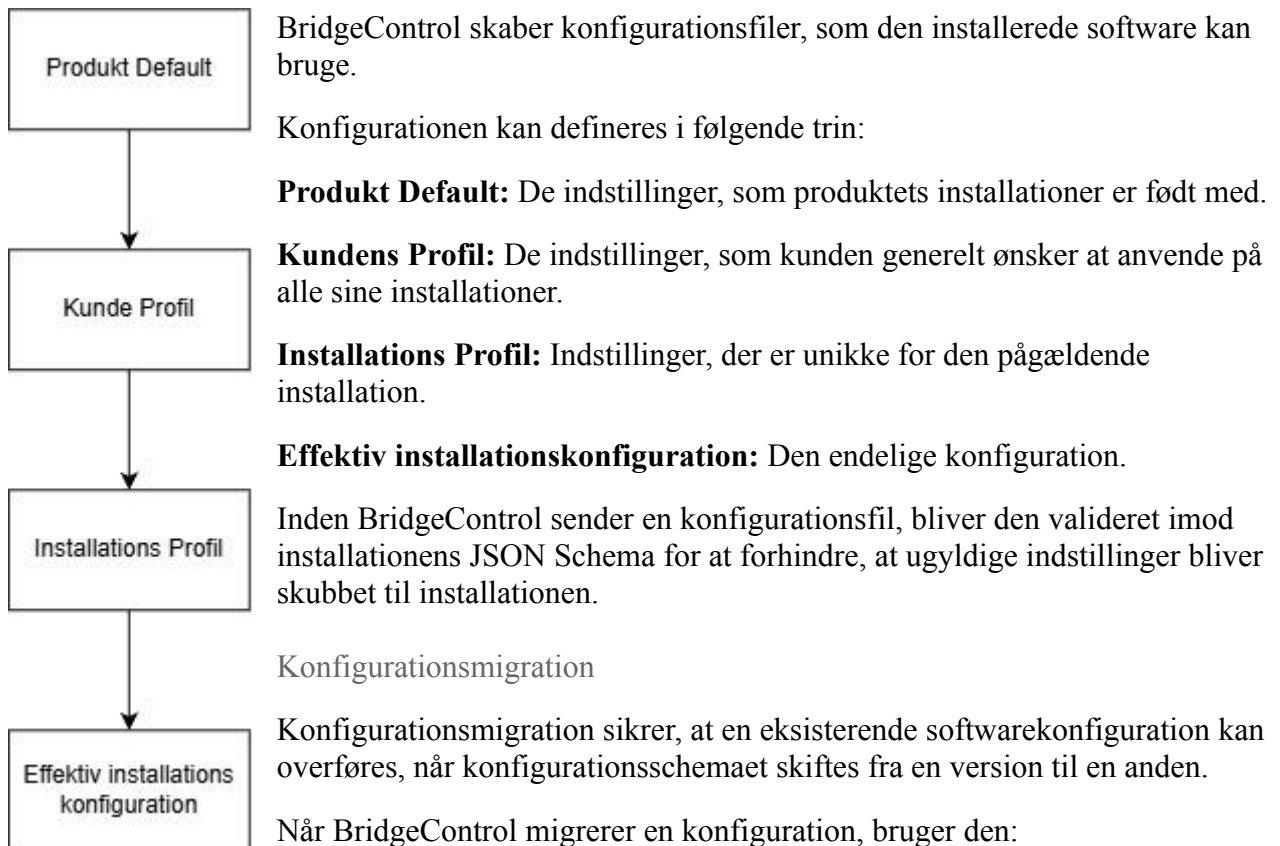
En distributør udvikler et produkt og registrerer dets version. Hver version indeholder:

- En MSI-pakke, der bruges til at installere eller opdatere programmet.
- En JSON Schema-fil, der beskriver konfigurationerne i forhold til den givne version.

Distributøren kan så tildele produkter af en bestemt version til kunder eller specifikke installationer.

Agenten downloader MSI-pakken, kører den lokalt og rapporterer resultatet til BridgeControl.

Konfigurationsstyring



Konfigurationsmigration

Konfigurationsmigration sikrer, at en eksisterende softwarekonfiguration kan overføres, når konfigurationsschemaet skiftes fra en version til en anden.

Når BridgeControl migrerer en konfiguration, bruger den:

- De eksisterende konfigurationer.
- Den gamle versions JSON Schema.
- Den nye versions JSON Schema.
- De migrationsregler, der er defineret mellem dem.

Eksisterende værdier gemmes, inden de kortlægges til deres tilsvarende værdier i det nye schema. Migrationsregler kan omdøbe, flytte, konvertere, introducere nye og slette værdier, der ikke længere understøttes.

Inden en migration valideres den eksisterende konfiguration mod dens eget JSON Schema. Efter en migration valideres den migrerede konfiguration mod det nye JSON Schema. Dette skal ske, inden konfigurationen bliver sendt til installationen.

En migration må ikke slette konfigurationsdata, uden at den aktive bruger godkender det. I logfilen skal det noteres, hvilke data der er blevet slettet, og hvem der har slettet dem.

Alle eksisterende værdier skal altså tages med til den nye konfigurationsfil, transformeres eller fjernes med en brugers godkendelse som en del af migrationen.

Dato: 27-09-2026

Skole: TEC

Hvis BridgeControl ikke kan afgøre, hvordan en værdi skal migreres, skal migrationen fejle og kræve afklaring frem for at producere en fejlagtig eller ufuldstændig konfiguration.

Brugere og adgang

BridgeControl bruger både roller og tilladelser:

BridgeControl Administrator: Håndterer distributører og BridgeControls egen platform.

Distributør Administrator: Håndterer brugere og kunder inden for sin egen virksomhed.

Kunde Administrator: Håndterer brugere og installationer inden for sin egen virksomhed.

Distributører og kunder skal være isolerede enheder. Det vil sige, at en distributør ikke kan få adgang til en anden distributørs data, og at en kunde ikke kan få adgang til en anden kundes data.

Overstyring og overvågning

Distributøren modtager et centralt overblik over:

- Egne kunder og deres installationer.
- Installerede og ønskede softwareversioner.
- Onlinestatus.
- Tidspunkt for seneste kommunikation.
- Konfigurationsstatus og revision.
- Licensstatus.
- Implementeringsstatus og historik.
- Succesfulde og fejlede installationer og opdateringer.
- Fejl og logs.

Typisk workflow

1. En distributør udvikler et produkt.
2. Distributøren uploader en ny versions MSI-fil, JSON Schema og default settings.
3. Produktet og licensen tildeles en kunde.
4. BridgeControl-Agenten installeres hos brugeren.
5. En installation registreres gennem BridgeControl-Agenten.

Dato: 27-09-2026

Skole: TEC

6. Distributøren vælger den ønskede version og konfiguration.
7. BridgeControl validerer og skaber den nye konfiguration.
8. Agenten downloader og kører MSI-filen.
9. Agenten tilføjer konfigurationerne.
10. Agenten rapporterer den installerede version, konfiguration og status.

Kort sagt er BridgeControl kontrolcenteret, der forbinder softwaredistributører med den software, de har installeret hos deres kunder. Det automatiserer programimplementering og konfigurationer uden at forsøge at styre hele kundens computer og operativsystem.

Problemformulering

I dette afsnit beskrives projektets overordnede problemstilling samt de tilhørende underspørgsmål.

Softwareforhandlere har flere produkter, som anvendes af forskellige kunder. Installation, konfiguration og opdatering af produkterne kræver manuel håndtering og bliver derfor mere tidskrævende, efterhånden som antallet af kunder og installationer vokser.

BridgeControl skal være en central platform, hvor softwareforhandlere kan administrere deres egne produkter og kunder. Systemet skal gøre det muligt at styre, installere, konfigurere, opdatere og overvåge installationerne.

Hvordan kan BridgeControl udvikles som et universelt software management system?

1. Hvordan kan BridgeControl struktureres, så flere softwareforhandlere kan anvende samme system, hvor forhandlerne og deres kunder kun har adgang til egne produkter og installationer?
2. Hvordan kan BridgeControl håndtere central og individuel konfiguration af software, så ændringer kan distribueres til de relevante installationer?
3. Hvordan kan installation og opdatering af programmer automatiseres, så software hos kunderne kan holdes opdateret uden manuel installation på den enkelte enhed?
4. Hvordan kan BridgeControl give softwareforhandleren et overblik over installationernes aktuelle version, konfiguration, status og licens?
5. Hvordan kan kommunikationen mellem BridgeControl og de enkelte installationer sikres, så uvedkommende ikke kan få adgang til programmer, konfigurationer eller kundedata?

Kravspecifikation

I kravspecifikationen definerer vi BridgeControls scope, funktionalitet og kvalitetskrav. Den beskriver brugerroller, access permissions, software deployment, configuration management og overvågning.

Vi giver også en ramme for systemets interfaces, datamodeller, sikkerhedskrav og acceptkriterier. Vi opstiller ligeledes implementeringsmæssige milepæle og identificerer beslutninger, der skal træffes, inden produktionen kan accepteres.

Funktionelle krav og User Stories

De funktionelle krav til BridgeControl er organiseret i otte milestones, som hver repræsenterer et centralt funktionsområde i systemet. De tilhørende User Stories beskriver de konkrete funktionelle behov inden for hvert milestone og fremgår af **Bilag 1 – User Stories**.

Milestone 1 – Foundation

Etabler sikker authentication, account management samt roles og permissions. Gør det muligt at administrere vendors, customers og users med kontrolleret adgang til ressourcer. Isolér hver vendors data i en separat database, håndhæv authorization og bevar historiske oplysninger.

Milestone 2 – Software Catalog

Opbyg et software catalog med products, product versions, validerede MSI installers og JSON Schemas. Kontroller deployment readiness, beskyt udgivne artifacts, administrer customer assignments og license entitlements, og vælg den ønskede product version for hver installation.

Milestone 3 – Agent Connectivity

Muliggør sikker installation, registration og authentication af Agents ved hjælp af beskyttede private keys. Bevar installation identity, understøt reconnection og decommissioning, og rapporter heartbeats, softwareversioner, configuration state, deployment results samt relevante errors og logs til BridgeControl.

Milestone 4 – Configuration Management

Generer en valideret Effective Configuration ved at kombinere Product Version Default, Customer Profile og Installation Profile efter faste merge rules. Gem configuration revisions centralt, distribuer kun gyldige og aktuelle indstillinger, og synliggør revision history, configuration drift og concurrent edit conflicts.

Dato: 27-09-2026

Skole: TEC

Milestone 5 – Configuration Migration

Overfør Customer Profiles og Installation Profiles sikkert mellem product versions gennem eksplícitte migration rules. Valider migration paths og resultater, kræve godkendelse ved destructive migration, bevar eksisterende configuration ved fejl, og opret én sporbar target configuration revision.

Milestone 6 – Software Deployment

Bring installationer i deres desired state gennem sikker levering af MSI installers og kontrolleret deployment. Håndter offline Agents, afbrydelser og retries sikkert, verificer actual state, og gem deployment history med separate resultater for MSI execution og configuration application.

Milestone 7 – Monitoring

Vis dashboards med installationernes operational state, connectivity, product versions, configuration revisions, license status og deployment outcomes. Fremhæv installationer, der kræver opmærksomhed, og vis relevante errors og logs, mens authorization begrænser oplysningerne til brugerens tildelte resource scope.

Milestone 8 – Audit & Hardening

Gør administrative og operationelle handlinger sporbare gennem audit logs og bevaret historik. Understøt sikre database schema upgrades samt separat backup og restore for hver vendor. Verificer authorization, concurrency controls, idempotency, lifecycle behavior og recovery på tværs af systemet.

Brugervejledning

Agent

Du skal have kundedgang, de nødvendige rettigheder og administratorrettigheder i Windows. Under Operations → Agents skal du vælge Download Windows Agent og derefter Enroll Agent. Vælg Customer, og generér en kode. Installer MSI-filen som administrator. Åbn Setup BridgeControl Agent, indtast HTTPS-serveradressen (kræver et betroet certifikat), vælg valgmulighed 1, og indtast koden. Kontrollér kunde, maskine og Heartbeats.

Produkt

Opret et produkt under Products, og vælg derefter Versions & artifacts → New version → Create draft. Upload MSI-filen, JSON Schema og standardfilen appsettings.json. Under Application deployment skal du vælge Desktop eller Service og angive den fulde sti til konfigurationsfilen samt enten den fulde sti til den eksekverbare fil eller navnet på Windows-tjenesten. Validér standardværdierne, gem, og vælg Publish version.

Dato: 27-09-2026

Skole: TEC

Licens

Vælg Assign product på Agenten, og angiv Product og Installation name. Opret en licens til kunden og produktet under Operations → Licenses, og angiv License reference, Valid from samt eventuelt udløbsdato og kapacitet. Under installationens Desired & reported state skal du vælge Assign license og tildele en gyldig licens med ledig kapacitet.

Konfiguration og installation

Under Administration → Configuration skal du vælge Customer override eller Installation override samt produkt og version. Redigér via Form eller JSON, vælg Save revision, og kontrollér Effective configuration. Installationsværdier tilsidesætter kundeværdier, som tilsidesætter standardværdier; lister erstattes. Under Set desired state skal du vælge Published version og den tilhørende Effective revision. Vælg derefter Request desired state. Følg forløbet under Deployments, og kontrollér, at den rapporterede version og revision stemmer overens med de ønskede. Gem dit arbejde i skrivebordsprogrammet før opdateringer.

Overvågningsoversigten / Installationernes tilstand

Overvågningsoversigten viser installationernes aktuelle tilstand: Agentens forbindelse, aktualiteten af den seneste rapport samt status for konfiguration, licens og deployment. Den opdateres automatisk hvert 30. sekund og fremhæver installationer, der kræver opmærksomhed.

Installation	Connectivity	Freshness	Configuration	License	Deployment
d757f956	Offline	Stale	Ready	Eligible	Reconciled

Offline 1 Configuration failures 0 Drift detected 0 Failed deployments 0

Connectivity – forbindelsen til installationens Agent

Online: Aktiv Agent med kontakt inden for 5 minutter.

Stale: Seneste kontakt er mellem 5 og 15 minutter gammel.

Offline: Agenten er inaktiv, eller den seneste kontakt er over 15 minutter gammel.

Unknown: Agenten eller oplysninger om seneste kontakt mangler.

Freshness – aktualiteten af installationens seneste rapport

Fresh: Rapporten er højst 15 minutter gammel og har en kendt status.

Stale: Rapporten er over 15 minutter gammel.

Unknown: Rapporten mangler eller har status Unknown.

Dato: 27-09-2026

Skole: TEC

Configuration – klargøring af installationens effektive konfiguration

Ready: En effektiv konfigurationsrevision er tilgængelig; den er ikke nødvendigvis installeret endnu.

Pending: Konfigurationen afventer behandling.

Failed: Behandlingen af konfigurationen mislykkedes.

Unknown: Der findes endnu ingen effektiv konfigurationsrevision.

License – installationens licensberettigelse

Eligible: Licenskravene er opfyldt.

InstallationUnavailable: Installationen, kunden eller produktet er inaktivt.

AgentUnavailable: Agenten mangler eller er inaktiv.

LicenseMissing: Ingen licens er tildelt.

LicenseUnavailable: Licensen er inaktiv, endnu ikke gyldig, udløbet eller tilhører en anden kunde eller et andet produkt.

LicenseCapacity: Antallet af aktive installationer overstiger licensens kapacitet.

Deployment – installationens seneste deployment

None: Ingen deployment findes.

Pending: Afventer udførelse eller et nyt forsøg.

Running: Agenten udfører deploymenten.

AwaitingState: Afventer en installationsrapport, der bekræfter resultatet; genstart kan være nødvendig.

Reconciled: Den rapporterede installationstilstand er bekræftet at matche den ønskede.

Failed: Deploymenten mislykkedes efter de tilladte forsøg.

Cancelled: Deploymenten blev annulleret.

Superseded: Deploymenten blev erstattet af en nyere ønsket tilstand.

Ved fejl skal du undersøge **Deployments** og **Logs**.

Teknisk produktdokumentation

I dette afsnit beskriver vi den tekniske opbygning og implementering af BridgeControl. Der gennemgås blandt andet systemarkitektur, teknologier, datamodeller, API og adgangskontrol samt håndtering af konfiguration, udrulning og overvågning.

Formål og systemarkitektur

BridgeControl er udviklet til central administration af softwareinstallationer hos flere softwareleverandører og deres kunder. Løsningen samler produkter, versioner, konfiguration, licenser, udrulning og overvågning, så en administrator kan angive den ønskede tilstand for en installation og følge, om den bliver opnået.

Dato: 27-09-2026

Skole: TEC

Vi har opdelt løsningen i en webgrænseflade, en central backend og en Windows-Agent. Webgrænsefladen bruges til administration, mens backend behandler forespørgsler, håndhæver regler og tilgår databaser og filarkiver. Agenten kører på kundens computer og udfører installation, opdatering og konfiguration af den administrerede software. Både webgrænsefladen og Agenten kommunikerer med backend via HTTPS. Hverken webgrænsefladen eller Agenten tilgår databaser eller filarkiver direkte.

Samspillet mellem komponenterne fremgår af figur 1, »System context and external actor interaction«, i bilaget BridgeControl Diagrams. Vi har desuden udarbejdet bilaget BridgeControl HTTP API Documentation, som beskriver API-kontrakten med endpoints, datamodeller, adgangskrav og fejlsvær. Dokumentets afsnit 1–3 beskriver arkitekturens fælles principper.

Software og teknisk driftsmiljø

Backend er implementeret i C# med ASP.NET Core på .NET 10. Entity Framework Core bruges til datatilgang og skemamigreringer i SQL Server. Strukturerede data overføres som JSON, og JSON Schema bruges til at kontrollere konfigurationsværdier. Backend anvender desuden SHA-256 til kontrolsummer og genererer en OpenAPI-beskrivelse af API'et.

Webgrænsefladen er udviklet med React, Material UI og Vite. React JSON Schema Form genererer konfigurationsformularer ud fra produktets JSON Schema. Validering i browseren hjælper brugeren, men backend kontrollerer også oplysningerne, før de accepteres.

Agenten er en separat Windows-service udviklet i .NET 10, som kører under LocalSystem. Det giver de nødvendige lokale rettigheder til at håndtere MSI-pakker, Windows-services og konfigurationsfiler. Agenten understøtter både serviceprogrammer og desktopprogrammer. Den starter ikke en desktopgrænseflade i servicekontoens session.

Systemet kræver ikke specialbygget hardware. Serveren skal kunne afvikle backend og SQL Server samt håndtere filarkiver. Den maskine, hvor Agenten kører, skal anvende Windows og have plads til downloads og administreret software. Minimumskrav til CPU, RAM og lager er ikke fastlagt; dimensioneringen afhænger af antallet af Agenter, pakkestørrelser og historik.

HTTPS-certifikatet skal passe til serverens DNS-navn eller IP-adresse og være betroet af klientmaskinen. En manuelt accepteret advarsel i browseren er ikke tilstrækkelig for Agentservicen. Agentens HTTPS-adresse og port skal være korrekte, og netværk og firewall skal tillade forbindelsen.

Datamodeller og adskillelse af leverandører

En softwareleverandør er systemets tenant, dvs. den overordnede grænse for adskillelse af leverandørdata. Platformdatabasen indeholder blandt andet leverandørregistrering, platformadministration og oplysninger til at finde leverandørens database og filarkiv. Hver leverandør har en separat database med egne kunder, brugere, produkter, installationer og driftsdata.

Dato: 27-09-2026

Skole: TEC

Backend vælger leverandørkonteksten ud fra den godkendte identitet. Et leverandørnavn ved login bruges til at finde den relevante leverandør, men er ikke i sig selv en adgangstilladelse. Almindelige leverandørafgrænsede forespørgsler kan derfor ikke skifte tenant ved blot at indsende et andet vendorId. Adskillelsen er vist i diagramdokumentets figur 3, »Vendor database routing and isolation«.

Bilagets tre ER-diagrammer beskriver henholdsvis platformdatabasen, leverandørens brugere og adgang samt produkter og konfiguration. Primærnøgler identificerer poster, og fremmednøgler beskriver relationer mellem dem. Brugerroller kobles til tilladelser gennem relationstabeller, mens konfigurationsrevisioner knyttes til produkter, versioner og de installationer, de vedrører.

Store filer opbevares i leverandørens filarkiv. Databasen indeholder metadata som filnavn, størrelse, kontrolsum og en intern reference til indholdet. Klienterne får adgang gennem kontrollerede API-endpoints frem for direkte arkivstier. ER-diagrammerne er ikke en fuldstændig gengivelse af alle implementeringsdetaljer. API-dokumentationens afsnit 28 beskriver supplerende behov for blandt andet Agentregistrering og sessionshåndtering.

HTTP API og adgangskontrol

API'et er versionsopdelt under /api/v1. Datafelter anvender camelCase, identifikatorer overføres som GUID'er, og tidsstempler angives i UTC. Lister er som udgangspunkt sideopdelte med 25 poster og højst 100 poster pr. side. Det begrænser størrelsen af det enkelte svar; klienten skal hente flere sider, når hele samlingen er nødvendig.

GET bruges til læsning, PUT til at sætte en tilstand, PATCH til delvise ændringer og POST til oprettelse og kommandoer. Identiske PUT-forespørgsler skal ikke skabe dubletter. Fejl returneres i Problem Details-format. API'et beskrives via OpenAPI på /openapi/v1.json og Swagger på /swagger. Se HTTP API-bilagets afsnit 3 og 24–27.

Brugere autentificeres med adgangstokens, og backend kontrollerer både identitet, tilladelse og adgang til den konkrete ressource. Platformadministratorer administrerer leverandører, mens leverandøradministratorer arbejder inden for deres egen tenant. Teknikere begrænses til tildelte kunder, mens kundebrugere begrænses til deres tilladte installationer. En rolle giver dermed ikke automatisk adgang på tværs af leverandører eller kunder. Modellen fremgår af diagramdokumentets figur 2 og API-dokumentationens afsnit 4, 6 og 23.

Agenten har sin egen identitet og bruger ikke en almindelig brugers login. Den registreres med en kortlivet enrollment-kode og får derefter legitimationsoplysninger til at hente Agenttokens. Backend begrænser dens handlinger til de installationer, der er tildelt den. Registrering og kommunikation beskrives i HTTP API-bilagets afsnit 18–21.

Konfiguration og versionsstyring

Et produkt kan have flere versioner. MSI- og schemafilerne for en publiceret version må ikke udskiftes ubemærket efter publicering. Installationer kan dermed knyttes til en bestemt release. Produktets versionsbetegnelse og MSI-pakkens interne versionsnummer er særskilte oplysninger.

Konfiguration opbygges i tre niveauer: produktets standardværdier, kundens tilpasninger og installationens tilpasninger. Backend beregner den effektive konfiguration ved at sammenflette niveauerne i denne rækkefølge. Objekter sammenflettes rekursivt, mens skalarer og arrays erstattes af det mere specifikke niveau. Null-værdier i objekttegenskaber fjernes fra det pågældende lag, så en overstyring kan ophæves og arvede værdier igen anvendes.

De indsendte lag og den samlede konfiguration valideres mod schemaet for målversionen. Resultatet gemmes som en ny revision med referencer til de anvendte kilderevisioner og en kontrolsum. Historikken gør det muligt at spore, hvilke værdier der dannede grundlag for en installation. Forløbet er vist i figur 4 og 8 i diagramdokumentet og beskrevet i HTTP API-bilagets afsnit 11–12.

Hvis en ny version kræver en ændret konfigurationsstruktur, anvendes migrationsregler mellem produktversionerne. Resultatet kan forhåndsvises og skal valideres mod målversionens schema. Destruktive regler kræver udtrykkelig godkendelse, og uafklarede værdier må ikke føre til en ufuldstændig konfiguration. Se figur 7 i diagramdokumentet og API-dokumentationens afsnit 13.

Udrulning og Agentens arbejde

Vi skelner mellem ønsket tilstand og faktisk rapporteret tilstand. Den ønskede tilstand angiver produktversion og effektiv konfigurationsrevision. Agentens rapport angiver den tilstand, der er observeret lokalt. Tildeling af et produkt er derfor ikke det samme som en gennemført installation.

En udrulning forudsætter blandt andet korrekt ejerskab, en egnet publiceret version, gyldig konfiguration og en gældende licens. Agenten henter tildelte installationer og ventende arbejde fra API'et. Før arbejdet udføres, kontrolleres den aktuelle tilladelse, og downloadede filer verificeres, herunder med SHA-256. Agenten udfører de nødvendige installations- eller konfigurationstrin og rapporterer resultatet tilbage.

Et gennemført forsøg er ikke alene bevis for, at den ønskede tilstand er opnået. Systemet sammenholder også rapporteret version, konfigurationsrevision, kontrolsum og status. For en service anvendes Running, mens Ready for et desktopprogram betyder, at installation, programfil og konfiguration er på plads. Ready beviser ikke, at en bruger har åbnet programmet eller indlæst de nye indstillinger.

Agenten gemmer lokal tilstand og igangværende arbejde for at håndtere afbrydelser uden blindt at gentage installationer. Fejl bevares, og genforsøg er begrænsede. Hvis en genstart er påkrævet, afventer Agenten genstart frem for selv at genstarte computeren. Se figur 5 og 6 i diagramdokumentet samt HTTP API-bilagets afsnit 14–17 og 19–21.

Overvågning og belastningsberegning

Agenten sender som standard et heartbeat hvert 60. sekund og kontrollerer normalt installationsarbejde med et interval på 30 sekunder. Heartbeat-funktionen kører uafhængigt af installationsarbejdet. Derfor kan backend fortsat modtage livstegn, mens Agenten arbejder på en MSI-pakke.

Dato: 27-09-2026

Skole: TEC

Den gennemsnitlige heartbeat-belastning kan estimeres som $R = N / T$, hvor N er antallet af Agenter, og T er intervallet i sekunder. Ved 100 Agenter og $T = 60$ bliver belastningen cirka 1,67 heartbeat-forespørgsler pr. sekund eller 6.000 pr. time. Beregningen omfatter kun heartbeats; opslag, downloads, tilstandsrapporter og genforsøg kommer oveni. Den er et dimensioneringseksempel og ikke en måling af maksimal kapacitet.

En kørende Windows-service er ikke nødvendigvis online set fra backend. Overvågningen bygger på modtagne livstegn og deres alder. Eksempelvis kan en forkert serveradresse eller et ikke-betroet certifikat forhindre kommunikation, selv om servicen er startet. Driftslogs beskriver hændelser og fejl, mens auditdata dokumenterer aktør, handling og resultat. Sammenhængene fremgår af figur 9 i diagramdokumentet og HTTP API-bilagets afsnit 22.

Testrapport

I dette afsnit beskriver vi, hvordan BridgeControl testes for at kontrollere systemets funktionalitet og identificere fejl. Der gennemgås både automatiserede og manuelle tests samt de anvendte testmetoder.

Testmetoder og testresultater

Backend Tests: 115 of 115 passed

Agent Tests: 74 of 74 passed

Vi anvender automatiserede tests til at kontrollere systemets funktionalitet og opdage fejl ved ændringer i koden. Backend og Agent testes med xUnit.

Automatiserede tests kan gentages med de samme forudsætninger og giver derfor en ensartet kontrol. De kan dog kun kontrollere de scenarier, der er beskrevet i testene. Beståede tests er derfor ikke i sig selv dokumentation for, at hele systemet er fejlfrit.

Unit tests kontrollerer afgrænsede dele af systemet. En test opstiller et kendt udgangspunkt, udfører en handling og sammenligner resultatet med det forventede. Det kan eksempelvis være at kontrollere, om en adgangskode overholder kravene, om en ugyldig konfiguration bliver afvist, eller om konfigurationsværdier bliver behandlet korrekt. Vi tester både gyldige input, ugyldige input og grænsetilfælde.

Unit tests køres løbende under udviklingen, især efter ændringer i den funktionalitet, de dækker. Ved fejlrettelser bør der tilføjes en test, som reproducerer fejlen og kontrollerer rettelsen. Testen kan derefter opdage, hvis den samme fejl opstår igen.

Integrationstests undersøger, om flere dele af systemet fungerer korrekt sammen. Backendens integrationstests bruger midlertidige SQL Server LocalDB-databaser og tester blandt andet API-kald, adgangskontrol og lagring af data. Det gør det muligt at kontrollere samspillet mellem API, forretningslogik og database uden at bruge produktionsdata.

Dato: 27-09-2026

Skole: TEC

Integrationstests køres ved ændringer, der påvirker samspillet mellem systemets dele, og inden en version betragtes som klar til aflevering. Manuelle tests gennemføres, når en brugerrettet funktion er klar, samt som en del af den afsluttende kontrol af løsningen.

Manuel test bruges til at vurdere brugergrænsefladen og de samlede arbejdsgange. Her gennemføres konkrete scenarier, eksempelvis login, redigering af konfiguration og håndtering af fejlmeddelelser. Formålet er at undersøge, om løsningen fungerer forståeligt og hensigtsmæssigt for brugeren.

Projektplanlægning

I dette afsnit beskriver vi, hvordan projektet planlægges og organiseres gennem projektperioden. Der gennemgås blandt andet den valgte udviklingsmetode, Product Backlog, User Stories, den foreløbige tidsplan og den løbende opgavestyring.

1. Formål

Formålet med projektplanen er at skabe et overblik over, hvordan projektet forventes at blive gennemført.

Da projektet udvikles efter en agil arbejdsmetode, betragtes projektplanen som foreløbig. Planen vil derfor blive opdateret løbende, efterhånden som projektet udvikler sig, og vi får et bedre overblik over opgaver, krav og tidsforbrug.

2. Udviklingsmetode

Vi har valgt at arbejde efter en agil udviklingsmetode med inspiration fra Scrum.

Projektet opdeles i mindre opgaver og funktioner, som prioriteres og udvikles løbende. Dette gør det muligt at tilpasse projektet, hvis der opstår nye krav, udfordringer eller ændringer under udviklingen.

Vi anvender Trello som vores Scrum/Kanban-board til at organisere og følge projektets opgaver.

Boardet er opdelt i:

- **Product Backlog** – Opgaver og funktioner, som skal eller muligvis skal udvikles.
- **To Do** – Opgaver, der er udvalgt til at blive arbejdet på.
- **In Progress** – Opgaver, der arbejdes på.
- **Testing/Review** – Opgaver, der skal testes eller gennemgås.
- **Done** – Færdige opgaver.

Trello-boardet opdateres løbende gennem projektperioden.

Dato: 27-09-2026

Skole: TEC

3. Product Backlog og milestones

Projektets krav og planlagte funktionalitet er samlet i en Product Backlog i Trello.

Product Backlog fungerer som et overordnet overblik over, hvad der forventes at blive udviklet i projektet.

Backloggen er på nuværende tidspunkt organiseret i otte overordnede milestones. En oversigt over projektets Product Backlog fremgår af **Bilag 1 – User Stories**.

Hver milestone indeholder funktioner og User Stories, som relaterer sig til det pågældende område af systemet.

Milestones bruges dermed til at opdele det samlede projekt i mindre og mere overskuelige områder.

Product Backlog angiver primært, **hvad der skal udvikles**, og skal derfor ikke i sig selv betragtes som en angivelse af den aktuelle arbejdsstatus.

Backloggen kan løbende ændres og omprioriteres gennem projektperioden.

4. User Stories

Projektets funktionelle behov er beskrevet gennem User Stories, som danner grundlag for funktionerne og opgaverne i vores Product Backlog.

User Stories vedligeholdes løbende i takt med projektets udvikling og ændringer i krav eller prioriteringer.

Den samlede oversigt over projektets User Stories fremgår af **Bilag 1 – User Stories**.

5. Foreløbig tidsplan

Den foreløbige tidsplan opdeler projektet i overordnede projektfaser og beskriver de aktiviteter, der forventes gennemført i hver fase. Da vi arbejder agilt, kan rækkefølgen og omfanget af aktiviteterne ændres undervejs i projektperioden.

Projektfase	Planlagte aktiviteter
Opstartsperiode	Idé, projektafgrænsning og krav
Planlægning	User Stories, Use Cases og Product Backlog
Design	Systemdesign, arkitektur, database og relevante diagrammer

Dato: 27-09-2026

Skole: TEC

Udvikling – iteration 1	Udvikling af grundlæggende funktionalitet
Udvikling – iteration 2	Udvikling af centrale funktioner
Udvikling – iteration 3	Yderligere funktioner, integration og forbedringer
Test	Test, fejlrettelser og kvalitetssikring
Afslutning	Dokumentation, færdiggørelse og klargøring til aflevering

De konkrete datoer og opgaver fastlægges og justeres løbende i takt med projektets udvikling og prioriteringen af Product Backlog.

6. Opgavestyring

Trello anvendes til den daglige opgavestyring gennem projektperioden.

Hver større funktion eller User Story kan opdeles i mindre Trello-opgaver. Opgaverne prioriteres og flyttes på boardet i takt med deres status, så vi løbende kan følge projektets fremdrift.

7. Løbende evaluering og ændringer

Da vi arbejder agilt, forventer vi, at projektplanen løbende vil blive justeret gennem projektperioden.

Vi vil løbende evaluere:

- Projektets fremdrift
- Product Backlog
- Prioriteringen af opgaver
- Tidsestimater
- Tekniske udfordringer
- Eventuelle nye eller ændrede krav

Projektplanen, Trello-boardet og User Stories kan derfor blive opdateret gennem projektperioden.

Dette giver mulighed for at tilpasse udviklingen på baggrund af de erfaringer, vi får undervejs.

Dato: 27-09-2026

Skole: TEC

Arbejdsfordeling

I projektets første uge udarbejdede Jonas vores Use Cases, kravspecifikation og User Stories, mens Amein planlagde, hvordan projektet skulle udvikles. Amein klargjorde desuden dokumenter, planlagde arbejdet med logbogen, udarbejdede projektplanen og fastlagde retningslinjer for kommunikationen under projektet.

Sprint 0

I uge 2 gennemførte vi Sprint 0, som havde til formål at skabe en klar ramme for vores API og dets endpoints. Dette gjorde vi for at kunne arbejde mere selvstændigt og mindske risikoen for, at manglende kommunikation senere skulle skabe problemer.

I Sprint 0 havde vi følgende tasks på vores Kanban-board:

- Define the system architecture
- Create vendor database ER Diagram
- Create BridgeControl Platform ER Diagrams
- Define the API
- Define Authentication/Authorization flows

Uge 3 og 4

Jonas arbejdede primært med vores API og Agent.

Amein arbejdede primært med vores Web UI.

Uge 5

Vi arbejdede sammen på at integrere UI, API og Agent og sikre, at systemets dele fungerede sammen. Amein arbejdede desuden med UX- og UI-design.

Begrundelse for metodevalg og teknologi

I dette afsnit begrundes vi vores valg af udviklingsmetode og de centrale teknologier, der anvendes i projektet. Valgene tager udgangspunkt i projektets behov, vores eksisterende kompetencer og muligheden for at udvikle og vedligeholde løsningen.

Agil udviklingsmetode

Vi har valgt at arbejde efter en agil udviklingsmetode, da projektets størrelse og kompleksitet gør det vanskeligt at fastlægge alle aspekter af løsningen på forhånd. Selvom Jonas tidligere har arbejdet

Dato: 27-09-2026

Skole: TEC

med de anvendte teknologier til både frontend og backend, har vi behov for løbende at kunne justere projektet og træffe beslutninger, efterhånden som nye behov og udfordringer opstår.

Derudover har vi begge erfaring med agile arbejdsmetoder og mindre erfaring med vandfaldsmodellen. Den agile tilgang giver os derfor mulighed for at arbejde ud fra en metode, vi allerede kender, samtidig med at projektet kan tilpasses gennem udviklingsperioden.

Backend

Vi har valgt C# til backenden, fordi sproget egner sig til projektets omfang, og fordi vi har god erfaring med det. Det giver os mulighed for at bygge videre på vores eksisterende kompetencer og fokusere på projektets funktionalitet.

Vi anvender .NET 10, som er en Long Term Support-version (LTS). Den lange supportperiode giver et stabilt grundlag for vedligeholdelse og videreudvikling af systemet.

Frontend

Til frontend har vi valgt React, Material UI (MUI) og React JSON Schema Form (RJSF). Jonas har tidligere erfaring med disse teknologier, og valget gør det derfor muligt at udnytte gruppens eksisterende kompetencer og samtidig anvende teknologier, der understøtter projektets behov.

React bruges til at opbygge brugergrænsefladen, mens MUI leverer færdige komponenter, som understøtter et ensartet design. Derudover anvender vi RJSF til at generere formularer ud fra JSON-schemaer og vise dem med MUI-komponenter. Det er særligt relevant for vores konfigurationsværktøjer, da formularerne kan tilpasses de enkelte applikationers konfigurationsmuligheder uden at skulle udvikles manuelt for hver version af hver applikation.

Projektdagbog/logbog

Gennem projektperioden har vi ført en løbende logbog, hvor vi for hver projektdag har dokumenteret vores arbejdsplan, de udfordringer vi har mødt, samt status på det udførte arbejde. Logbogen giver dermed et kronologisk overblik over projektets faktiske udvikling.

I projektets første fase arbejdede vi blandt andet med problemformulering, kravspecifikation, User Stories, Use Cases og planlægning af projektet. Herefter blev arbejdet mere teknisk med blandt andet systemarkitektur, ER-diagrammer, API-design og Authentication/Authorization. Senere i projektperioden blev arbejdet fordelt mellem udviklingen af API, Agent og Web UI, hvorefter fokus i højere grad blev rettet mod integrationen mellem systemets forskellige dele.

Logbogen viser også, hvordan projektets udfordringer har påvirket arbejdet undervejs. Under frontendudviklingen opstod der eksempelvis problemer med at opsætte og forbinde backenden til frontenden. Det medførte, at vi måtte arbejde tættere sammen om integrationen, før udviklingen af Web UI kunne fortsætte med den nødvendige forbindelse til API'et.

Dato: 27-09-2026

Skole: TEC

Da projektet er udviklet efter en agil arbejdsmetode, er logbogen samtidig med til at synliggøre forskellen mellem det planlagte og det faktiske projektførløb. Den viser, hvilke opgaver der reelt blev arbejdet med på de enkelte tidspunkter, hvilke udfordringer der opstod, og hvordan arbejdet blev tilpasset undervejs. Logbogen danner derfor også grundlag for afsnittet Realiseret tidsplan, hvor projektets faktiske forløb sammenholdes med den foreløbige planlægning.

Det fulde projekt dagbog/logbog fremgår af **Bilag 2 - Logbog**.

Realiseret tidsplan

Den realiserede tidsplan er udarbejdet på baggrund af vores logbog og beskriver, hvordan projektførløbet faktisk udviklede sig. Tidsplanen kan dermed sammenholdes med den foreløbige tidsplan og viser de ændringer og afvigelser, der opstod undervejs i projektet.

Projektet blev gennemført med en agil tilgang, og arbejdsopgaverne blev derfor løbende tilpasset projektets udvikling. Frontend og backend blev desuden udviklet parallelt, mens dokumentation og rapportskrivning foregik sideløbende med udviklingen.

24.-28. august – Projektstart og projektafgrænsning

Vi blev introduceret til H5-projektførløbet og dannede projektgruppen. Da der ikke var et konkret projektforslag fra Jonas' arbejdsplads, valgte vi at arbejde videre med BridgeControl som projekttidé.

Vi arbejdede herefter med casebeskrivelse, problemformulering, User Stories, diagrammer og gruppesamarbejdsaftale. Vi oprettede også et Trello Kanban-board til styring af projektets opgaver. I perioden modtog vi desuden undervisning i vandfaldsmodellen og agile udviklingsmetoder.

31. august-7. september – Kravspecifikation og projektplanlægning

User Stories og de første diagrammer blev færdiggjort, hvorefter kravspecifikationen og den foreløbige projektplan blev udarbejdet.

En stor del af perioden fra 31. august til 4. september blev anvendt på undervisning i Big Data 2, hvilket begrænsede den tid, vi kunne anvende på BridgeControl. Kravspecifikationen og den foreløbige projektplan blev efterfølgende færdiggjort og sendt til vores vejleder med henblik på feedback.

8.-11. september – Systemdesign og teknisk planlægning

Vi definerede systemarkitekturen og opsatte strukturen for projektrapporten. Herefter arbejdede vi med ER-diagrammer for blandt andet Vendor Database og BridgeControl Platform Database.

Vi definerede desuden API'et, authentication- og authorization-flowet samt kommunikationen mellem BridgeControl Agent og API'et.

Dato: 27-09-2026

Skole: TEC

14.-20. september – Udvikling af backend, Agent og frontend

Udviklingsarbejdet blev opdelt, så Amein primært arbejdede med frontend, mens Jonas primært arbejdede med backend og Agent.

Amein undersøgte frontendens design, oprettede et separat Trello-board til frontendopgaverne og opsatte React, Node, GitHub samt de nødvendige libraries. Frontendudviklingen blev herefter påbegyndt med den grundlæggende applikationsstruktur og navigation.

Jonas opsatte C#/.NET-projektet, oprettede Models og DTO'er og arbejdede med blandt andet authentication, authorization og Vendor-funktionalitet. Jonas gennemførte de planlagte backend-sprints og udviklede en BridgeControl Agent.

21.-23. september – Færdiggørelse af frontend og integration

De centrale management views i frontenden blev færdiggjort, herunder Dashboard, Customer Management, Installation Management, Product Management og User Management. Arbejdet med Authentication API Integration var næsten færdigt, men den sidste del var afhængig af forbindelsen til backend.

Der opstod problemer med at opsætte backenden på Ameins computer. Vi valgte derfor at mødes fysisk for at løse problemerne sammen. Backend blev herefter sat op på Ameins computer, så den kunne forbindes med frontenden.

Den 23. september blev frontenddelen færdiggjort, mens Jonas fortsatte med at forbedre backenden.

24.-25. september – Rapportskrivning og afslutning

Fokus blev flyttet fra den primære udvikling til rapportskrivning og færdiggørelse af projektet. Vi arbejdede videre med rapporten, foretog rettelser og opdaterede logbogen, mens der samtidig blev foretaget mindre ændringer i backenden.

Den 25. september modtog vi den sidste feedback fra vores vejleder inden afleveringen. Herefter var planen at anvende afleveringsweekenden på at færdiggøre rapporten og refaktorere koden inden aflevering.

Afvigelser fra den foreløbige tidsplan

Den foreløbige tidsplan beskrev projektføreløbet som en række overordnede faser bestående af opstart, planlægning, design, udviklingsiterationer, test og afslutning. Den realiserede tidsplan viser, at projektet i praksis blev mindre sekventielt end først planlagt.

En væsentlig afvigelse var, at arbejdet med BridgeControl i perioden fra 31. august til 4. september var begrænset på grund af undervisningen i Big Data 2. Derudover påvirkede sygdom projektarbejdet på enkelte dage.

Dato: 27-09-2026

Skole: TEC

Under udviklingsfasen blev arbejdet opdelt mellem frontend og backend. Amein arbejdede primært med frontend, mens Jonas arbejdede med backend og BridgeControl Agent. De forskellige dele af systemet blev derfor udviklet parallelt frem for som tre klart adskilte udviklingsiterationer.

Integrationen mellem frontend og backend krævede også mere tid end forventet. Frontenden kunne ikke færdiggøres fuldstændigt, før backenden kunne køres og forbindes på Ameins computer. Den 21. og 22. september blev der derfor brugt ekstra tid på opsætning og fejlfinding, hvorefter forbindelsen mellem frontend og backend kunne etableres.

Rapportskrivningen foregik samtidig med udviklingsarbejdet og blev intensiveret i projektets afsluttende fase. Den oprindelige interne deadline for rapporten måtte justeres. Samtidig blev den officielle afleveringsdeadline forlænget til søndag, hvilket gav yderligere tid til færdiggørelse af rapporten, feedback, rettelser og refaktorering.

Samlet set fulgte projektet den overordnede retning fra den foreløbige tidsplan, men de enkelte aktiviteter overlappede i højere grad end først forventet. Dette stemmer overens med vores agile arbejdsform, hvor planlægningen og prioriteringen løbende blev tilpasset projektets aktuelle status og de udfordringer, der opstod undervejs.

Konklusion

Projektet viser, at et generelt software management system kan udvikles ved at kombinere en central platform med en lokal Agent og fælles beskrivelser af produkternes installation og konfiguration. BridgeControl samler dermed administrationen af produkter, kunder og installationer og giver et grundlag for at reducere den manuelle håndtering, når antallet af installationer vokser.

Flere softwareforhandlere kan anvende samme platform gennem adskillelse af forhandlernes data og adgangskontrol baseret på roller og tilknytninger. Det begrænser brugernes adgang til de produkter, kunder og installationer, som de har rettigheder til at administrere.

Central og individuel konfiguration håndteres ved at kombinere fælles indstillinger med tilpasninger til den enkelte kunde og installation. JSON-schemaer beskriver produkternes konfigurationsmuligheder og bruges til at validere indstillingerne og generere formularer. Den samlede konfiguration beregnes centralt og hentes af Agenten, som anvender den på den relevante installation.

Installation og opdatering automatiseres gennem Agenten på kundens enhed. Når Agenten er installeret og tilknyttet platformen, kan den hente installationsfiler og udføre de tildelte installationer og opdateringer. Dermed reduceres behovet for manuel installation af de understøttede produkter på hver enkelt enhed.

Forhandlerens overblik skabes ved at samle installationernes rapporterede version, konfiguration og status med de tilknyttede licensoplysninger. Oplysningerne gør det muligt at følge installationernes tilstand og identificere afvigelser mellem den ønskede og den rapporterede tilstand. Overblikkets aktualitet afhænger af Agenternes seneste tilbagemeldinger.

Kommunikationen mellem platformen og installationerne beskyttes gennem krypterede forbindelser, autentifikation og adgangskontrol. Agenter identificeres særskilt, og deres adgang begrænses til de installationer samt tilhørende programmer og konfigurationer, som de er tilknyttet. Disse mekanismer begrænser risikoen for uvedkommendes adgang til systemets data.

BridgeControl besvarer dermed problemstillingen med en fælles arkitektur, der kan håndtere forskellige softwareprodukter gennem de samme administrative arbejdsgange. Systemet er dog ikke universelt i absolut forstand, da ikke alle programmer og operativsystemer understøttes uden yderligere tilpasning.

Løsningens anvendelighed afhænger endvidere af, at produkterne følger de understøttede installations- og konfigurationsformer. Projektet demonstrerer således en fleksibel platform for central softwareadministration, mens driftssikkerhed og tidsbesparelser ved større udrulninger kræver yderligere afprøvning i realistiske driftsmiljøer.

Refleksion

BridgeControl kan videreudvikles til at understøtte flere platforme og softwaretyper. Mulige udvidelser omfatter:

- **Agenter til flere operativsystemer:** Den nuværende Windows-Agent kan suppleres med Agenter til eksempelvis ChromeOS, Linux og macOS. Understøttelse af macOS og ChromeOS kræver særskilte løsninger, der tager højde for platformenes begrænsninger for installation og administration.
- **Flere installations- og pakkeformater:** Understøttelsen kan udvides til eksempelvis EXE-, DMG- og JAR-filer. Det kræver forskellige metoder til installation, opdatering og kontrol af resultatet, da formaterne ikke følger samme installationsproces.
- **Flere konfigurationsformater:** Ud over JSON kan systemet udvides til at understøtte eksempelvis XML og INI. Det kræver understøttelse af læsning, skrivning og validering af hvert format. XML kan valideres med XSD-schemaer, mens eksempelvis INI kræver særskilte regler for konfigurationens struktur og indhold.
- **Automatisk opdatering af Agenten:** Agenten kan i fremtiden udvides med mulighed for automatisk at opdatere sig selv, så nye versioner kan distribueres uden manuel opdatering på kundens enhed.
- **Styring af installationer gennem API'et:** API'et kan udvides med endpoints til at starte og stoppe en installation.
- **Localization:** Brugergrænsefladen er på nuværende tidspunkt hardcoded på engelsk. I fremtiden kan der implementeres localization, så brugergrænsefladen kan understøtte flere sprog.

Dato: 27-09-2026

Skole: TEC

- **Temaer:** Brugergrænsefladen understøtter på nuværende tidspunkt kun dark mode. I fremtiden kan den udvides med eksempelvis light mode, dark mode, et accessibility-tema for svagtseende samt automatisk valg af tema baseret på brugerens systemindstillinger.
- **Configuration Migration:** Da projektet var større end enlig forventet, så var der ikke tid til at teste configuration migration og vi er derfor ikke sikre på at det faktisk virker.

Disse udvidelser vil gøre BridgeControl anvendeligt til flere produkter og driftsmiljøer samt forbedre brugeroplevelsen og tilgængeligheden af systemet.