

Bygg for alle
(Universell Utforming)
Systemdokumentasjon

1 Endringslogg

Dato	Forfatter	Endringsbeskrivelse
20.09.2011	Asbjørn Willersrud	Lagt til info om Jetty/Derby for registreringsmodulen.
14.09.2015	Øyvind Marthinsen	Oppdatert teknologier og lagt til del om rapportering
06.02.2020	Jørgen Sirhaug	Endret bilde og beskrivelse for Publikumsmodul, samt lagt til avsnitt om AWS.
09.04.2026	Synnøve Halle	Oppdatert teknologiseksjoner: Java 8, Tomcat 7, MySQL 8.4.7 LTS, jQuery 3.5.0 (npm), AWS. Lagt til seksjoner for bfa-publikum: Next.js, React 18, MUI v5, Emotion v11, Highcharts og Lingui. Rettet URL for publikumsmodul. Lagt til beskrivelse av Docker-basert deployment for bfa-publikum, inkludert nytt avsnitt om Docker.
16.04.2026	Synnøve Halle	Ny seksjon om autentisering
21.04.2026	Tor Johannessen	Ny seksjon om brukere og mobilnr

Innholdsfortegnelse

1	Endringslogg	2
2	Innledning	6
3	Funksjonell beskrivelse	7
3.1	Skjemaadministrasjon	7
3.2	Registrering	8
3.3	Statistikk og rapportering	9
3.4	Publikumsmodul	11
4	Teknologier	12
4.1	Java 8	13
4.2	Tomcat 7	13
4.3	Spring 1.2.8	13
4.4	Spring MVC	13
4.5	Hibernate 3.2.1	13
4.6	MySQL database	13
4.7	Acegi Security	14
4.8	Yahoo! UI Library	14
4.9	jQuery	14
4.10	Velocity template engine	14
4.11	JEP – Java Math Expression Parser	14
4.12	AppFuse	15
4.13	FitNesse / Selenium / JUnit	15
4.14	Git	15
4.15	Gradle/Apache Ant/Gulp	15
4.16	Jetty	15
4.17	Apache Derby	16
4.18	Amazon Web Services	16
4.19	API grensesnitt	16
4.20	Next.js	16
4.21	React 18	16
4.22	Material-UI / MUI v5	17
4.23	Emotion v11	17
4.24	Lingui	17
4.25	Docker	17
4.26	Highcharts	17

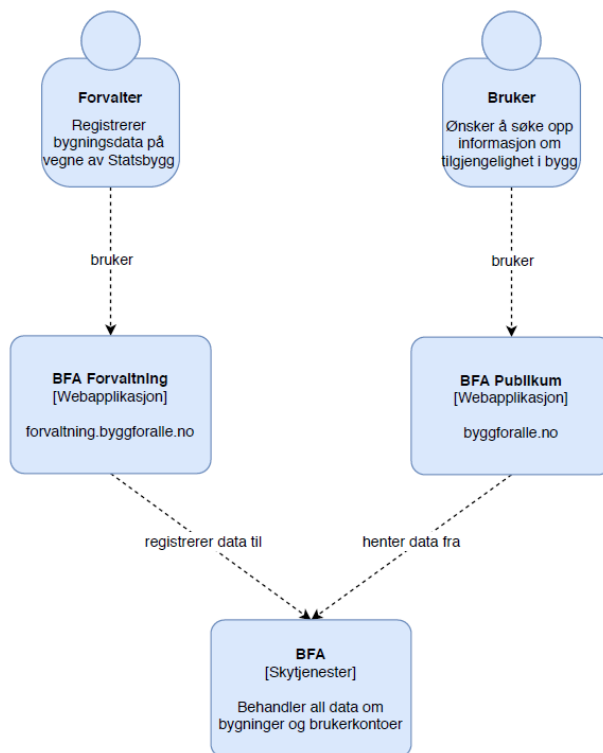
5	Prosjektstruktur Forvaltning	18
5.1	DAO	18
5.2	Service	18
5.3	Web	18
6	Datastruktur / domenemodell	19
6.1	Diagramnotasjon / Hibernate	19
6.2	Bygningsdata	20
6.2.1	Bygningselement / Status	20
6.2.2	Tiltak / FunksjonshemmingsGruppe / Gjennomføring	21
6.2.3	Attachment	21
6.2.4	Bygningstype/Eiendomstype	21
6.2.5	Måleverdi	21
6.3	Skjemadata	22
6.3.1	Bygningselementtype	22
6.3.2	Kategori	22
6.3.3	Målepunkt	22
6.4	Kriterier	23
6.4.1	KriterieProfil	23
6.4.2	Kriterium	23
6.4.3	Krav	23
6.4.4	KriterieProfilSubsett / KriterieProfilSubsettGruppering	23
6.5	Flerspråklighet	24
6.5.1	Lokalisering ved hjelp av ressursfiler	24
6.5.2	Lokalisering ved hjelp av databasetabeller	24
6.5.3	Oversetting til et nytt språk	25
6.6	Brukere og roller	25
6.6.1	Roller	25
6.6.2	Brukere	25
7	Autentisering og innlogging	26
7.1	Innloggingsflyt	26
7.2	Totrinnsautentisering (2FA)	26
7.3	Passord	26
7.3.1	Krav til Passord	26
7.3.2	Hashing og lagring	27
7.3.3	Passordutløp	27
7.3.4	Endre passord	27

7.4	Glemt passord / tilbakestilling	27
7.5	Ny bruker og verifikasjon av mobilnummer	27
7.6	Brukernavn og e-postadresse	27

2 Innledning

Bygg for alle (BFA) er et system laget for å registrere, forvalte og tilgjengeliggjøre (på web) informasjon om tilgjengelighet i offentlige bygninger. Systemet har fire hovedfunksjonsområder:

- Registrering av bygningsdata og tiltak knyttet til disse.
- Forvaltning av bygnings- og skjemadata (målepunkter og kriterier for Universell Utforming).
- Rapportering og statistikk av målepunkter, kriterier og tiltak for Universell Utforming.
- En søkemodul, hvor publikum kan søke på bygninger og besøksmål.



Systemet *Bygg for alle* er utviklet i Java og basert på en rekke open source rammeverk og biblioteker. Ajax-teknikker benyttes for å oppnå et rikt og responsivt brukergrensesnitt. Se kapittel 4 for en beskrivelse av teknologiene som er benyttet i løsningen, og kapittel 5 for en beskrivelse av oppbyggingen av systemet.

Datamodellen som ligger til grunn for løsningen er basert på en domenemodell som representerer bygningsdata i en generisk struktur. Se kapittel 6 for en beskrivelse av datamodellen.

3 Funksjonell beskrivelse

Systemet har to hovedområde:

- Forvaltning (forvaltning.byggforalle.no)
- Publikumssøk (<https://www.byggforalle.no>)

Systemet har fire hovedfunksjonsområder: skjemaadministrasjon, registrering, rapportering og publikumssøk.

Disse er beskrevet nedenfor.

3.1 Skjemaadministrasjon

Skjemaadministrasjonen definerer grunnlaget for registrering av bygningsdata:

- Den definerer hvilke typer bygningsselementer som skal kunne registreres (f eks dør, veistrekning/korridor, resepsjon etc.).
- Den definerer hvilke målepunkter som skal registreres for hvert enkelt bygningsselement (f eks "dørstokkens høyde", "passasjebredde" etc.).
- Den har støtte for å definere kriterier for hvert enkelt målepunkt, og å angi hvilke funksjonshemmingsgrupper kriteriet gjelder for (f eks "Passasjebredde minst 86 cm", gjelder for rullestolbruker).
- Den har også støtte for en tekstlig beskrivelse av kravene til hver bygningsselementtype.

Skjemaadministrasjonen er brukergrensesnittsmessig lagt opp med et treeview til venstre, hvor bygningsselementtyper, deltyper og målepunkter danner en trestruktur. Til høyre ser man et detaljview (et dynamisk generert skjema) som beskriver den valgte typen eller det valgte målepunktet.

Statsbygg

Registrering Skjemaadministrasjon Oversettelse Publikumsmodul Brukere Rediger profil Logg ut Hjelp

Passasjebredde : FlyttallMålepunkt

Navn:
 Obligatorisk: ☒
 Viser for publikum: ☒
 Form validator:
 MaxVerdi:
 MinVerdi:
 Enhet:

Kriterieprofil: Universell Utforming
 Kriterium:
 Forklaring:

☒ Rullestolbruker ☐ Svaksynt
☐ Annen bevegelseshemming ☐ Miljøhemmet
☐ Døv ☐ Blind
☐ Forståelseshemmet ☐ Tunghørt

Kriterieprofil: VTF
 Kriterium:
 Forklaring:

Feltene *Form validator*, *MaxVerdi* og *MinVerdi* er felter for å kontrollere hva som skal være lovlige input-verdier for de ulike målepunktene, for å minske sjansen for feiltastinger.

Kriterier uttrykkes ved hjelp av boolske uttrykk, og de brukes til å avgjøre om en måling oppfyller eller ikke oppfyller kriteriene for det angjeldende målepunktet. I tillegg er det støtte for å legge inn en forklaring av det boolske uttrykket – det er denne forklaringen som vises i publikumsmodulen (se eksemplet i figuren ovenfor: ≥ 86 er ”oversatt” til ”Minst 86”).

Man kan velge en eller flere funksjonshemmingsgrupper som hvert enkelt kriterie skal gjelde for.

Se kapittel 6.3 for en beskrivelse av datamodellen som ligger til grunn for skjemaadministrasjonen.

3.2 Registrering

I registreringsmodulen registreres besøksmål, barrierer og veistrekninger, og i tillegg opprettes strukturen over bygninger: eiendommer, regioner og institusjoner. Alt dette representeres i databasen som bygningselementer. Et bygningselement opprettes med en valgt type (som definert i skjemaadministrasjonen) og plasseres på valgt sted i treet.

The screenshot shows the Statsbygg web application interface. On the left is a tree view of building elements (Byggningsdata) under the 'Institusjon [Statsbygg]' node. The tree includes 'Region' (MIDT-NORGE, NORØ, SØR, TRØNDELAGE, VEST, ØST), 'Eiendom' (Anders Sandvigsst. 43, Lillehammer - F, Arkitektur- og designhøgskolen i Oslo), 'Bygning' (Arkitektur- og designhøgskolen i Oslo), 'Intervju', 'Inngang' (Hovedinngang), and 'Dør'. The 'Dør' node is expanded, showing 'Korridor eller vei [Til resepsjon]', 'Resepsjon [Resepsjon]', 'Korridor eller vei [Til kantine]', 'Korridor eller vei [Til 1M43 auditorium]', 'Korridor eller vei [Til verkstedsfløy]', and 'Korridor eller vei [Til parkering]'. The 'Resepsjon [Resepsjon]' node is selected. On the right is a form for 'Resepsjon' with fields for 'Navn' (Resepsjon), 'Kommentar', 'Status' (Uten publiseringsbegrensning), 'Diskens høyde' (120 cm), 'Plass foran disken' (checked), 'Bredde' (300 cm), 'Dybde' (300 cm), 'Rullestol får plass under bordet' (dropdown), 'Sittemulighet' (Ja), 'Visuelt kollapsystem' (dropdown), 'Visuelt alarmsystem' (dropdown), 'Teleslynge/skranseslynge' (dropdown), 'Belysning' (lux), and 'Belysning for hørselshemmede' (dropdown). The form also has buttons for 'Lagre', 'Kopier', 'Klipp ut', 'Lim inn', 'Slett', and 'Eksporter'.

Venstre del av skjermbildet viser treet av bygningselementer. Roten av treet er de ulike institusjonene (f eks Statsbygg), og treet forgrener seg ut i de ulike bygningene og bygningenes besøksmål.

Skjemaet til høyre genereres dynamisk ut ifra hvilket bygningselement man har valgt, og viser de relevante målepunktene. I tillegg ligger annen funksjonalitet i ulike faner i skjemaet (oppbygging av sti/trestruktur, vedlegg og tiltak). Er bygningselementet av kategori *bygning*, vil også funksjonalitet for tildeling av registrator og eksport av bygningsdata være tilgjengelig.

Registrerte data kan overføres mellom lokal installasjon og sentral server ved hjelp av eksport og import av xml-filer.

Se kapittel 6.2 for en beskrivelse av datamodellen som ligger til grunn for registreringsfunksjonaliteten.

3.3 Statistikk og rapportering

I registreringsmodulen er det også mulig å hente ut ulike rapporter. Hvilke rapporter man kan hente ut er avhengig av om man står på en bygning eller en institusjon. På institusjon er det mulig å hente ut statistikk/rapport for satsingbygg og en bygningsoversikt rapport.

[Institusjon \[Statsbygg\]](#)

Institusjon

KopierKlipp utLim innEkspander

RedigeringBygg opp stiVedleggTiltakKraaRapporter

Tiltaksrapport

Rapport for året:

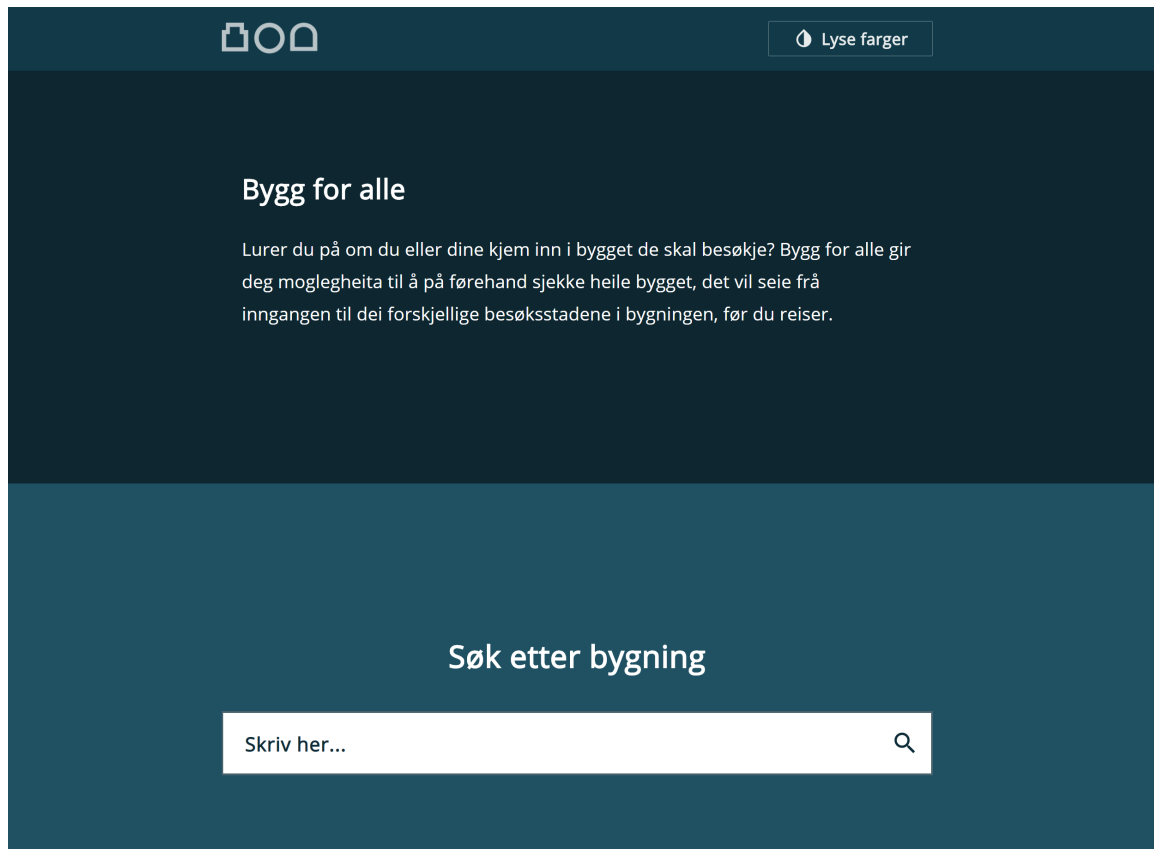
Statusrapport

Bygningsoversikt

For bygninger er det mulig å hente ut en liste over besøksmål med tilhørende sti til målet, tiltaksrapport, målepunktrapport, måleverdier for uoppfylte krav og en bygningsoversikt rapport.

3.4 Publikumsmodul

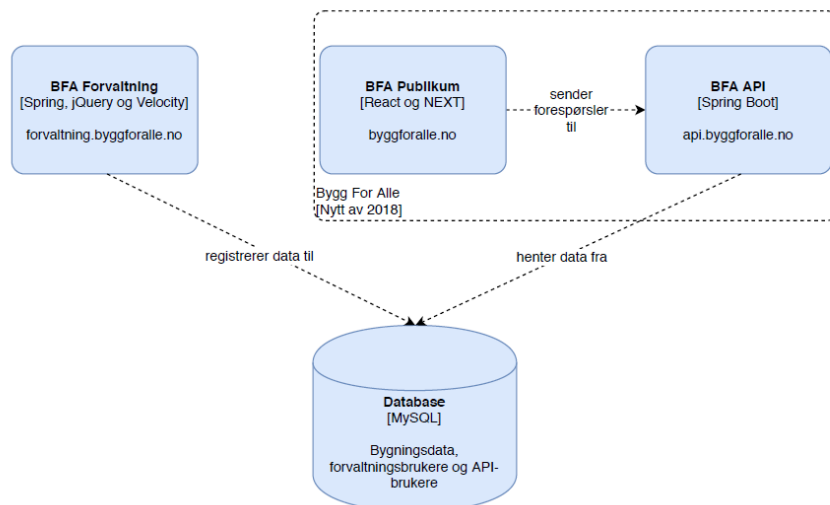
I publikumsmodulen (www.byggforalle.no) har publikum mulighet til å søke opp bygninger og å undersøke tilgjengeligheten til steder de er interessert i å besøke, og ble i 2018/-19 laget på nytt. Brukere kan søke etter bygninger og rom (besøksmål), og vil her finne beskrivelser og informasjon om disse. I tillegg vil brukere kunne se om det er mulige hindringer for dem ved et besøk. Applikasjonen er utviklet med Next.js og React, og benytter Material-UI (MUI) for komponenter, Emotion for styling og Lingui for internasjonalisering. Applikasjonen kjøres som en Docker-container på en EC2-instans i AWS, og deployes via AWS CodeDeploy.



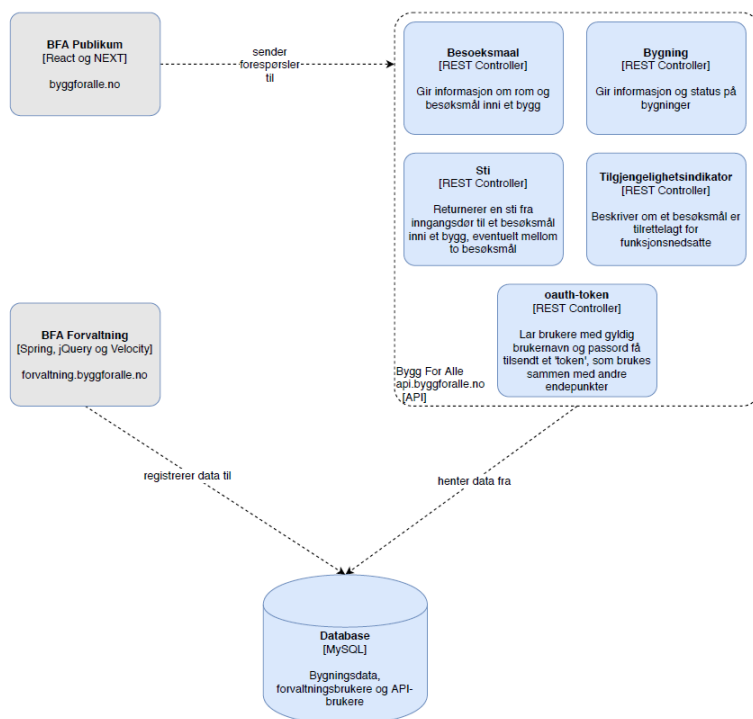
Publikumssidene er utviklet med tanke på tilgjengelighet for alle.

4 Teknologier

Løsningen er basert på en Java-plattform med utstrakt bruk av åpen kildekode programvare. Alle komponenter er kjente, velutprøvde, og er valgt med fokus på funksjonalitet og stabilitet. Tegningen under viser arkitekturen til den sentrale serveren som kjører forvaltningsmodulen. Publikumsmodulen (bfa-publikum) er en separat applikasjon basert på Node.js og Next.js, og kjøres som en Docker-container. Registratorene kjører en tilpasset versjon av samme applikasjon på en Jetty instans og med Apache Derby database.



Figur 1 Teknisk arkitektur som er benyttet i løsningen



Figur 2 API oversikt

4.1 Java 8

Java 8 med Open JDK 1.8.0 benyttes som programmeringsspråk for løsningen. Java-klassene som utgjør programlogikken kjører i en applikasjonstjener, og aktiveres fra en Java Servlet som også styres via URLer.

Lenke: <http://java.sun.com/>

4.2 Tomcat 7

Bygg for alle benytter Apache Tomcat som applikasjonstjener. Apache Tomcat er kjøremiljø for Java Servlets og Java Server Pages. Tomcat er åpen kildekode og referanseimplementasjonen av servlet container for Java Servlets. Merk: Tomcat benyttes kun av forvaltningsmodulen (bfa-forvaltning). Publikumsmodulen kjører ikke på Tomcat.

Lenke: <http://tomcat.apache.org/>

4.3 Spring 1.2.8

Spring er et lagdelt åpen kildekode Java applikasjonsrammeverk som støtter blant annet konfigurasjonsstyring, abstraksjonslag for databasetransaksjoner, integrasjon med Hibernate, og utvikling av web-applikasjoner. Spring blir i praksis limet som knytter alle programkomponentene sammen.

På persisteringssiden fungerer Spring som et overbygg som abstraherer mesteparten av kompleksiteten ved bruk av Hibernate. I tillegg brukes Spring til å styre transaksjoner deklarativt.

Lenke: <http://www.springframework.org/>

4.4 Spring MVC

Spring MVC er et rammeverk for utvikling av webapplikasjoner basert på Model View Controller tankegangen.

Lenke: <http://www.springframework.org/>

4.5 Hibernate 3.2.1

Hibernate er et åpen kildekode ORM (Object Relational Mapping) rammeverk. Hibernate støtter utvikling av persistente Java-klasser for databasetilgang, med egenskaper som assosiasjon, arv, polymorfisme og Java collections. Hibernate støtter også definisjon av databasespørringer i sin egen portable SQL utvidelse (HQL), i databasens egen SQL, eller ved hjelp av Javabaserte kriterier og eksempel-objekter.

Lenke: <http://www.hibernate.org/>

4.6 MySQL database

Løsningen bruker MySQL som databasemotor. All databasetilgang skjer gjennom Hibernate. Databaseversjonen ble i 2025/2026 oppgradert til MySQL 8.4.7 LTS på AWS RDS, og MySQL

Connector/J ble oppgradert til versjon 8.4.0 med moderne JDBC-driver (com.mysql.cj.jdbc.Driver).

Lenke: <http://www.mysql.com/>

4.7 Acegi Security

Acegi Security er et åpen kildekode rammeverk som tilbyr omfattende tjenester for autentisering og autorisering i forretningsapplikasjoner basert på Spring rammeverket.

Lenke: <http://acegisecurity.org/>

4.8 Yahoo! UI Library

Yahoo! User Interface Library er en samling rutiner og brukergrensesnittkontroller skrevet i JavaScript for bygging av rike, interaktive webapplikasjoner ved hjelp av teknikker som DOM skripting, asynkrone http kall, og annen AJAX.

Yahoo! UI Library brukes i Bygg for alle blant annet for tre-view, tab-view, dialoger, opplasting av filer via Ajax og for alle Ajax asynkrone kall.

Lenke: <http://developer.yahoo.net/yui/>

4.9 jQuery

jQuery er et bibliotek som gjør det enklere å jobbe med javascript. I nyutviklet funksjonalitet er dette ofte blitt foretrukket fremfor Yahoo UI og Prototype som er blitt brukt i eldre funksjonalitet. I 2025 ble jQuery oppgradert til versjon 3.5.0, og avhengigheten ble migrert fra bower_components til node_modules. På sider som også bruker Prototype.js benyttes jQuery.noConflict() for å unngå konflikter mellom bibliotekene.

4.10 Velocity template engine

Velocity er en Java-basert template engine som tilbyr et enkelt og effektivt sett med instruksjoner. Velocity gir tilgang til data fra Java objekter, ved å prosessere maler som inneholder fri blanding av data / innhold og instruksjoner til Velocity motoren.

I løsningen benyttes Velocity blant annet til å generere all html som hentes via Ajax kall.

Lenke: <http://jakarta.apache.org/velocity/>

4.11 JEP – Java Math Expression Parser

JEP er et Java bibliotek for å tolke og beregne matematiske uttrykk. JEP støtter brukerdefinerte variabler, konstanter og funksjoner, og kan dermed lett utvides med ønsket funksjonalitet.

I løsningen benyttes JEP for å validere om sett av måleverdier tilfredsstiller definerte kriterier.

Lenke: <http://www.singularsys.com/jep/>

4.12 AppFuse

AppFuse er et programvareprosjekt for kickstart av J2EE web-applikasjoner. AppFuse tillater utviklere å komme enkelt og raskt i gang med programvareteknologier som Spring, Hibernate og Spring MVC. Appfuse setter opp prosjektets katalogstruktur, med refererte jar-filer, build targets, og store mengder kildekode. Databasen genereres ved hjelp av xdoclet.

N.B. Dette er ikke blitt brukt nyere utvikling av systemet.

Lenke: <http://raibledesigns.com/wiki/Wiki.jsp?page=AppFuse>

4.13 FitNesse / Selenium / JUnit

FitNesse er en web-server og et testverktøy for software basert på Ward Cunninghams's Framework for Intergrated testing. FitNesse fokuserer på akseptansetesting. Testene kjøres i virkelige browsere ved hjelp av Selenium og Fit bibliotekene. FitNesse testene kan brukes som dokumentasjon på at brukstilfeller er implementert i henhold til beskrivelsene.

Selenium er et verktøy for testing av web applikasjoner. Testene kjøres direkte på forskjellige web browsere blant annet Internet Explorer, Firefox og Opera. Selenium kan kjøres på både Windows, Linux og Macintosh.

JUnit er et rammeverk for enhetstesting. Vi bruker JUnit til automatisert testing av lavnivå databasetilgang og tjenestelogikk.

N.B. Det er lenge siden disse testene er blitt vedlikeholdt

Lenker: <http://www.fitnesse.org>, <http://www.openqa.org/selenium>, <http://www.junit.org>

4.14 Git

Git brukes som versjonskontrollsystem. Et versjonskontrollsystem behandler filer og kataloger over tid. Et tre av filer er plassert i et sentralt depot (repository). Depotet ligner mye på en vanlig filserver, med det unntaket at det husker hver eneste forandring noensinne gjort på filene og katalogene. Dette tillater deg å hente fram eldre versjoner av dataene dine, eller studere historien for hvordan dataene dine har forandret seg.

4.15 Gradle/Apache Ant/Gulp

Gradle og Apache Ant er et verktøy for automatisk bygging av programvare. Hoveddelen av byggeprosessen baserer seg fortsatt på en Ant-byggefil i xml-format. Denne filen blir lest inn i Gradle og blir kjørt derfra. Planen er å gradvis bytte ut Ant med Gradle som hoved byggesystem.

I tillegg til Gradle er det tatt i bruk Gulp som et byggesystem for frontend kode.

4.16 Jetty

Registrator-modulen deployer samme applikasjon som kjører sentralt. Applikasjonen konfigureres som en registrator-installasjon og deploys på en Jetty server.

Jetty er en lettvekts webserver og ble valgt blant annet for å få en lett installasjonsprosess for nye registratorer.

Lenke: <http://jetty.codehaus.org/jetty/>

4.17 Apache Derby

Apache Derby er en lettvekts database implementert i Java og brukes i registrator-modulen. I likhet med Jetty ble Derby valgt for å gjøre installasjon- og vedlikeholdsprosessen så enkel som mulig for registratorer.

Lenke: <http://db.apache.org/derby/>

4.18 Amazon Web Services

I 2017 ble alle tjenester hos Bygg For Alle migrert over til AWS. Applikasjonsserverne kjører i dag som EC2-instanser (*Elastic Compute Cloud*, virtuelle servere kjørende hos Amazon), mens databasene, både for produksjon og test, kjører på RDS-servere (*Relational Database Service*). RDS-databasen kjører MySQL 8.4.7 LTS. S3-bøtten for vedlegg benytter serversidekryptering (AES256). Infrastructure as Code er definert i Terraform med separate miljøer for test og produksjon.

Lenker: <https://aws.amazon.com/>
<https://aws.amazon.com/ec2/>
<https://aws.amazon.com/rds/>

4.19 API grensesnitt

Oversikt over API i Bygg for alle

Lenker: <https://api.byggforalle.no/swagger-ui.html>

4.20 Next.js

Next.js er et React-basert rammeverk for serversiderendering (SSR) og statisk nettstedsgenerering. Publikumsmodulen (bfa-publikum) er bygget med Next.js og benytter Pages Router. I 2024/2025 ble applikasjonen oppgradert gjennom versjon 12, 14 og til 16.1.7, med bl.a. migrering fra det tredjeparts ruting-biblioteket next-routes til Next.js sitt innebygde rutesystem, og fra webpack 4 til webpack 5.

Lenke: <https://nextjs.org/>

4.21 React 18

React er et JavaScript-bibliotek for bygging av brukergrensesnitt. Alle klassekomponenter i bfa-publikum er migrert til funksjonskomponenter med React Hooks. Applikasjonen kjører nå på React 18 og krever Node.js 20 eller nyere.

Lenke: <https://react.dev/>

4.22 Material-UI / MUI v5

MUI (tidligere Material-UI) er et React-komponentbibliotek basert på Google Material Design. bfa-publikum bruker MUI v5/v6 for brukergrensesnittkomponenter som knapper, modaler og ikoner (@mui/material, @mui/icons-material). MUI ble introdusert som del av moderniseringen av applikasjonen.

Lenke: <https://mui.com/>

4.23 Emotion v11

Emotion er et CSS-in-JS-bibliotek som brukes for komponentbasert styling i bfa-publikum. Applikasjonen benytter @emotion/react, @emotion/styled og @emotion/server (for SSR).

Emotion ble oppgradert fra v9 til v11 som del av MUI-migreringen.

Lenke: <https://emotion.sh/>

4.24 Lingui

Lingui er et internasjonaliseringsbibliotek (i18n) for JavaScript og React. bfa-publikum bruker @lingui/core og @lingui/react for håndtering av oversettelser og lokaler. I 2024/2025 ble Lingui modernisert til v5 med useLingui()-hooken og lazy t-makroer, og lokal no ble normalisert til no-NB.

Lenke: <https://lingui.dev/>

4.25 Docker

Publikumsmodulen er pakket og deployes som et Docker-image bygget med en multi-stage Dockerfile basert på node:20-alpine. I byggsteget installeres avhengigheter og Next.js-applikasjonen kompiles. Det endelige imaget kjører server.js, som starter en Express-server med Next.js SSR.

Deployment til EC2 håndteres av AWS CodeDeploy ved hjelp av en appspec.yml-konfigurasjon. Ved hver deploy lastes det nye imaget inn og erstatter den kjørende containeren.

Lenke: <https://www.docker.com/>

4.26 Highcharts

Highcharts v11.4.8 brukes på siden utkvitterteRapport.jsp for å visualisere statistikk om ferdigstilte og ikke-ferdigstilte uu-bygg, dokumentert i seksjonen om [Statistikk og rapportering](#).

Lenke: <https://www.highcharts.com/>

5 Prosjektstruktur Forvaltning

Bygg for alle applikasjonen er delt i tre lag: DAO (Data Access Object), service og web.

5.1 DAO

DAO-laget består av domenemodellen og DAO klasser. Domenemodellen er i form av Java klasser annotert med xdoclet hibernate annotasjoner. Disse klassene/annotasjonene er kilden som databaseskjemaet og hibernate konfigurasjonen blir generert fra.

DAO klassene inneholder metoder for å lese, lagre og slette domeneobjekter fra databasen. Disse klassene er konfigurert som Spring beans og bruker Spring sitt Hibernate overbygg for Hibernate session håndtering slik at sessions ikke håndteres fra koden, men konfigureres gjennom Spring. Spring har i tillegg bedre exception håndtering enn Hibernate sine standard exceptions.

5.2 Service

Service-laget er et overbygg for DAO-laget og inneholder applikasjonslogikk-metoder og transaksjonshåndtering.

Hoveddelen av service-laget består av manager klasser som er et direkte overbygg for DAO klassene. Spring brukes til å konfigurere manager objekter og å koble disse til relevante DAO-objekter, dette gjøres ved hjelp av Spring dependency injection.

Metoder i manager klassene er konfigurert for å enten starte en ny transaksjon (hvis ingen fantes fra før) eller gjenbruke en eksisterende transaksjon. Transaksjonshåndteringen er konfigurert med Spring, koden forholder seg ikke til transaksjoner.

I tillegg til manager klasser med transaksjonshåndtering inneholder service-laget klasser som implementerer funksjoner utover vanlig les/skriv. Eksempler på slike er eksport/import av data til/fra XML format og validering av måleverdier mot kriterier.

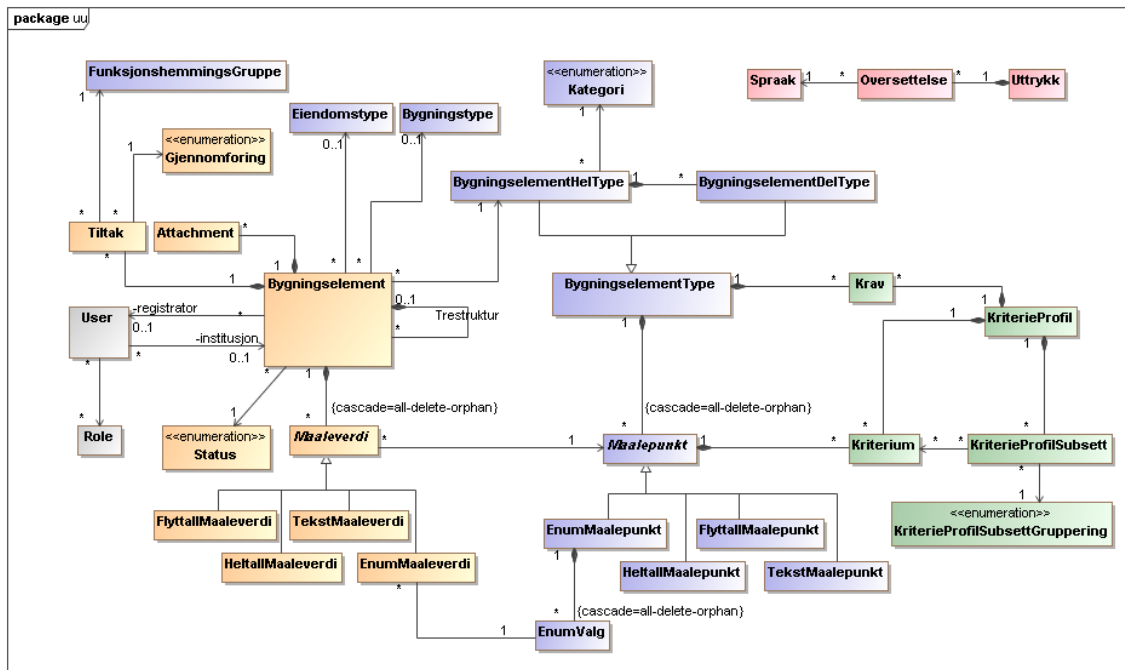
5.3 Web

Web-laget inneholder Java kontroller klasser, Velocity templates, jsp'er, tag library implementasjoner, statiske web-filer (javascript, css, bilder) og alle konfigurasjonsfilene for web-laget.

Kontroller klassene håndterer alle http requests, oversetter http parametere og kaller service metoder fra managere. Koblingen til manager objektene som trengs fra hver kontroller gjøres ved hjelp av Spring dependency injection. En kontroller returnerer enten html eller xml til klienten.

6 Datastruktur / domenemodell

Løsningen er basert på en domenemodell som representerer bygningsdata i en generisk struktur. Dette kapittelet dokumenterer domenemodellen. Fordi databasetabellene blir generert fra domeneklassene, er utgangspunktet for dokumentasjonen domeneklassene og ikke databasetabellene.



Figur 2 Oversikt over domenemodell

Den generiske strukturen datastrukturen muliggjøres av de fire sentrale klassene Bygningselement, BygningselementType, Målepunkt og Måleverdi. Hvert Bygningselement er av en BygningselementType, og denne bestemmer hvilke Målepunkter som skal registreres for dette Bygningselementet. For hvert Målepunkt registreres det så en Måleverdi.

6.1 Diagramnotasjon / Hibernate

Hver klasse i modellen representerer en tabell i databasen. Unntaket er <<enumeration>> klassene som bare tilsvarer en integer kolonne i tabellene hvor de refereres.

Hvert attributt i klassemodellen (ikke vist i oversiktsfiguren) representerer en kolonne i databasen. Assosiasjoner representerer vanligvis en fremmednøkkel, men kan også representere en koblingstabell ved mange til mange assosiasjoner.

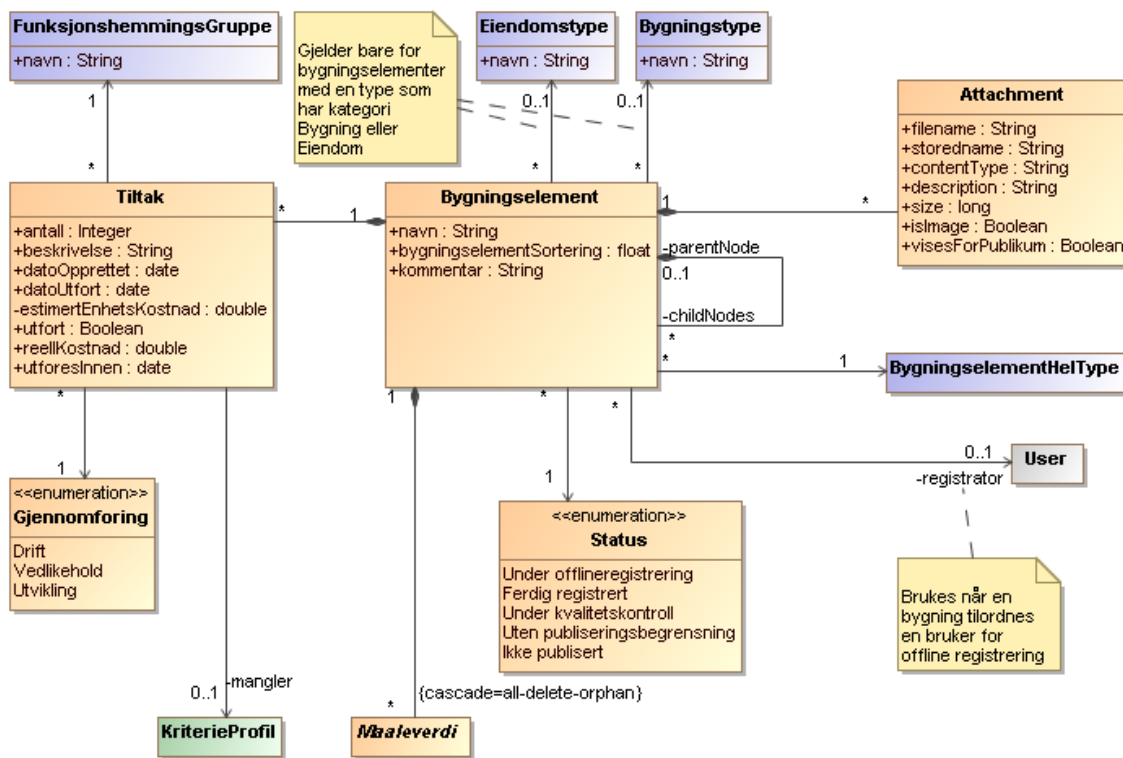
Komposisjonsassosiasjonene (fylte svarte firkanter) tilsvarer cascade=delete i Hibernate konfigurasjonen, om ikke det står noe annet. For bygningselement betyr dette at om et bygningselement slettes, så slettes også alle måleverdier, tiltak og attachments som eies av bygningselementet. I tillegg slettes alle bygningselementer som utgjør subtreet under dette bygningselementet (trestruktur assosiasjonen).

Arv relasjonene i diagrammet er implementert med Hibernate sin "Table per subclass" strategi. For måleverdihierarkiet vil for eksempel dette si at det finnes en tabell for den abstrakte klassen Måleverdi og en tabell for hver klasse som arver fra Måleverdi. Et objekt av typen FlyttallMåleverdi vil da representeres med en rad i Måleverdi tabellen og en rad i FlyttallMåleverdi tabellen, disse radene kobles sammen ved å ha lik primærnøkkel. For mer om dette se Hibernate dokumentasjonen.

(http://www.hibernate.org/hib_docs/v3/reference/en/html/inheritance.html)

Hver klasse har, i tillegg til attributtene som vises, en primærnøkkel og en UUID (Universally Unique Identifier). UUID brukes blant annet for å identifisere elementer på tvers av databaser ved import av bygninger fra offline registrering.

6.2 Bygningsdata



Figur 3 Detaljer om bygningsdata

6.2.1 Bygningselement / Status

Et bygningselement representerer en del av en bygning, selve bygningen eller en eiendom/institusjon. Det er instanser av denne klassen som bygger opp trestrukturen som vises i registreringsmodulen. Eksempler på bygningselementer er: "Dør til Kontor 3. etasje", "Arkitektur- og designhøgskolen i Oslo" og "HC WC 1. etasje". For bygningselementer som representerer bygninger (bygningselementer med en type som har kategori "Bygning") gjelder alle Status alternativene, mens for andre bygningselementer gjelder kun "Ikke publisert" og "Uten publiseringbegrensning". Et bygningselement med status "Ikke publisert" vil hindre alle underelementer å vises i publikumsmodulen.

6.2.2 Tiltak / FunksjonshemmingsGruppe / Gjennomføring

Tiltak representerer planlagt/gjennomført arbeid på et bygningsselement. Hvert tiltak kobles til en funksjonshemmingsgruppe for å beskrive hvilke for hvilken gruppe tiltaket er ment å gjøre bygget mer tilgjengelig.

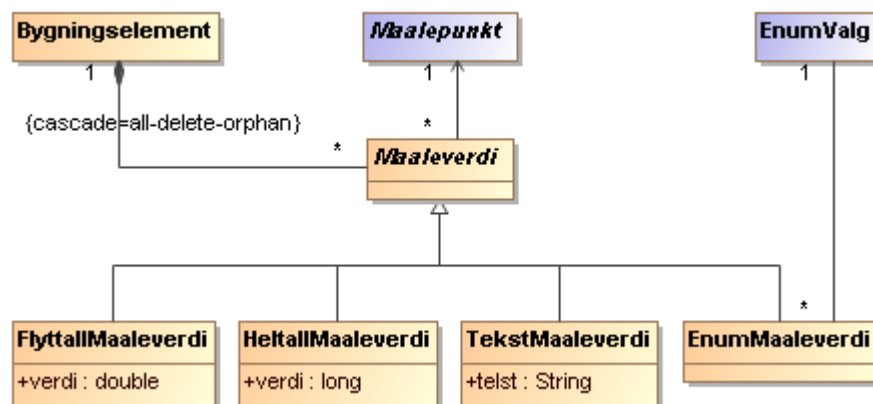
6.2.3 Attachment

Attachment inneholder en referanse til en fil som er lagt ved et bygningsselement. Hvis dette er et bilde vil det bli generert en komprimert (80x80) versjon av bilde som vises i brukergrensesnittet. Ved sletting av et attachment objekt vil også filen det refereres til slettes.

6.2.4 Bygningstype/Eiendomstype

Disse klassene brukes for å klassifisere bygninger og eiendommer. Det er bare bygningsselementer av kategori Bygning og Eiendom som skal ha bygnings- og eiendomstype.

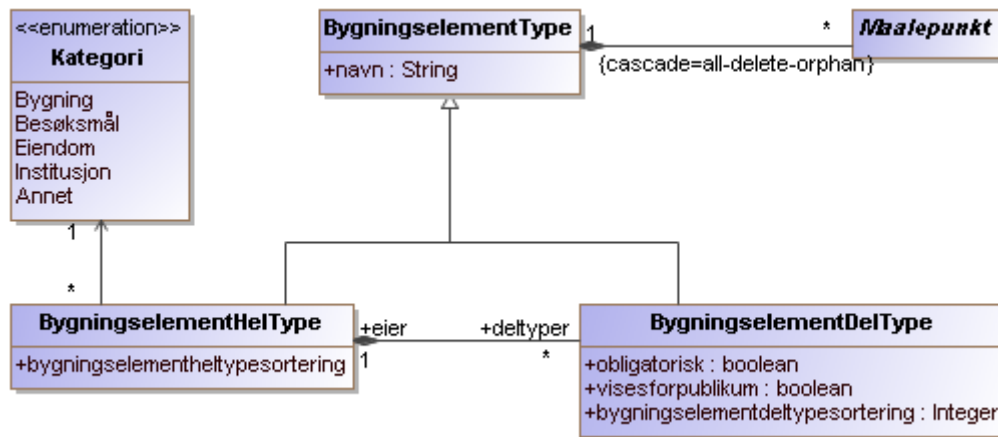
6.2.5 Måleverdi



Figur 4 Detaljer om måleverdi

De faktiske verdiene som registreres lagres som måleverdier. Når et bygningsselement lagres opprettes det en måleverdi for hvert målepunkt eid av typen til bygningsselementet. For eksempel hvis det opprettes et bygningsselement av typen "Dør" og "Dør" har målepunktet "Passasjebredde", vil det opprettes en måleverdi som knyttes til målepunktet "Passasjebredde" og det nye bygningsselementet.

6.3 Skjemadata



Figur 5 Detaljer om bygningselementtype

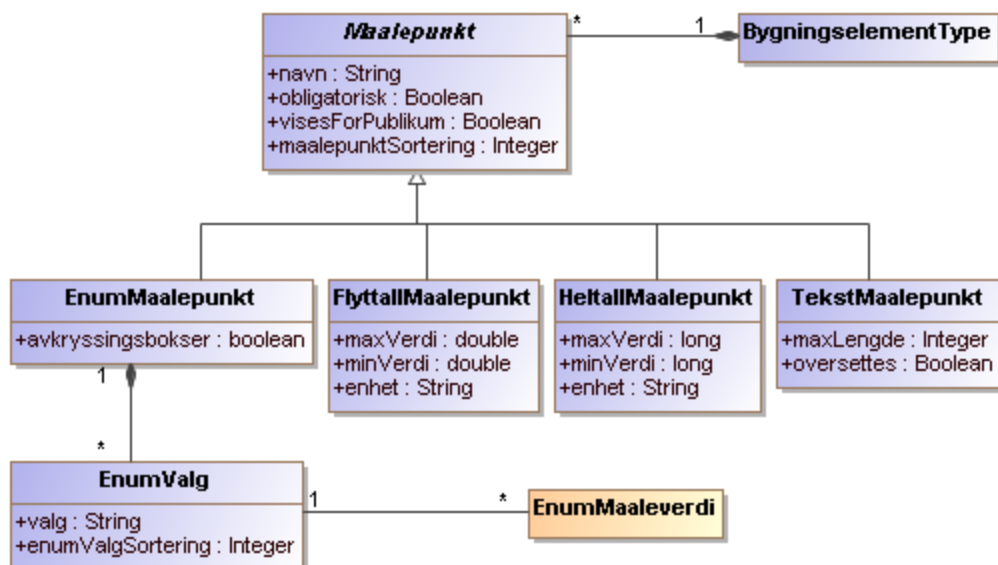
6.3.1 Bygningselementtype

Hvert bygningselement har en type, denne klassen representerer denne typen. Eksempler på bygningselementtyper er "Bygning", "Dør", "Rom", og "Heis". Hver bygningselementtype kan bestå av andre bygningselementdeltyper. Bygningselementtypen "Trapp" vil således bestå av blant annet bygningselementdeltypene "Rekkverk", "Markering" og "Belysning".

6.3.2 Kategori

Hver bygningselementheltype har en kategori. Kategorien "Annet" brukes på alle bygningselementtyper som er en del av en bygning (og som ikke er et besøksmål).

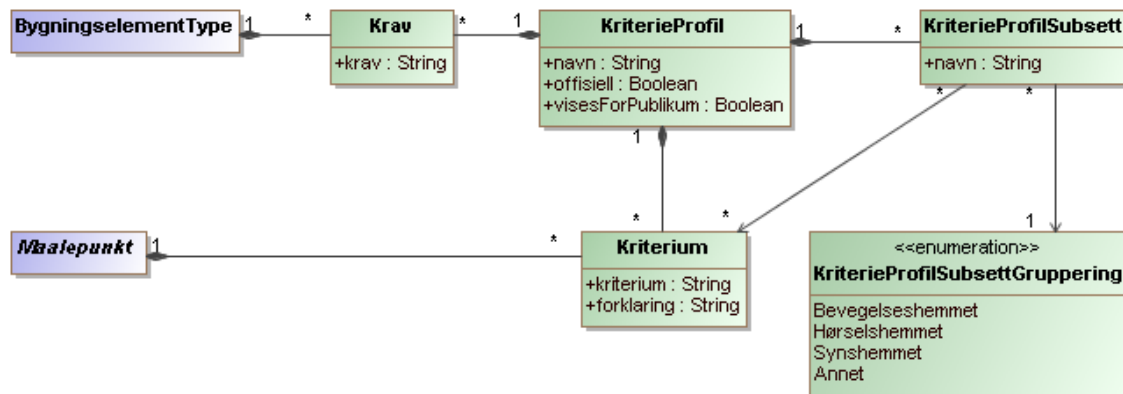
6.3.3 Målepunkt



Figur 6 Detaljer om målepunkt

Hver bygningselementtype har et sett med målepunkter som definerer hva som skal registreres for bygningselementer av denne typen. Målepunktene for en Trapp vil være for eksempel "Minste bredde", "Maks antall trinn etter hverandre", "Trinnenenes høyde" og "Trinnenenes dybde". Det finnes fire forskjellige typer målepunkt: flyttallmålepunkt, heltallmålepunkt, tekstmålepunkt og eummålepunkt/flervalgmålepunkt (Ja/Nei, God/Middels/Dårlig).

6.4 Kriterier



Figur 7 Detaljer om kriterier / kriterieprofil

6.4.1 KriterieProfil

En kriterieprofil er en samling kriterier. Kriterieprofilene som finnes i systemet er Universell Utforming og VTF. Offisiell feltet bestemmer om det skal genereres input felter for kriterier/krav for denne profilen i målepunkt/bygningselementtype skjemaene. VisesForPublikum feltet er ikke i bruk.

6.4.2 Kriterium

Kriterium knyttes til et målepunkt og en kriterieprofil. Kriterium feltet er et tekstlig uttrykk som tolkes automatisk og kan evaluere om en måleverdi oppfyller kriteriet. For eksempel kan et kriterium for målepunktet "Minste bredde" tilhørende bygningselementtypen "Trapp" være "> 90". Dette Kriteriet vil evaluere alle verdier for "Minste bredde" større enn 90 til oppfylt.

Forklaring feltet vises i publikumsmodulen istedenfor det logiske uttrykket som tolkes automatisk. For eksempel hvis kriterium feltet er "> 90" kan forklaring være "Må være større enn 90".

6.4.3 Krav

Krav er en tekstlig beskrivelse av kriteriene som gjelder for en bygningselementtype.

6.4.4 KriterieProfilSubsett / KriterieProfilSubsettGruppering

Et kriterieprofilsubsett inneholder referanser til et subsett av kriteriene for en kriterieprofil. Eksempler på kriterieprofilsubsett er "Døv", "Blind" og "Rullestolbruker". Dette brukes for å definere hvilke kriterier som gjelder for hvilken funksjonshemming.

KriterieProfilSubsettGruppering gjøre det mulig å vise kriterieprofilsubsettene i grupper i publikumsmodulen.

6.5 Flerspråklighet



Figur 8 Språk

Svært mange applikasjoner har behov for å fremstå i to eller flere språkdrakter, avhengig av brukernes nasjonalitet eller preferanser. Det finnes mange måter å støtte flerspråklighet på, og en av de vanligste metodene, lokalisering ved hjelp av ressursfiler, er allerede en del av det AppFuse genererte prosjektet vårt. Støtte for lokalisering av tall og datoformater finnes også innebygget, ved hjelp av AppFuse.

6.5.1 Lokalisering ved hjelp av ressursfiler

Tekstlig innhold separeres fra design ved å lagre nøkler og tekst i enkle tekstfiler kalt ressursfiler. Det finnes typisk en ressursfil per språkvariant. Hver ressursfil inneholder alle de samme nøklene, pluss den oversatte teksten som tilhører hver enkelt nøkkel. Eksempel fra en norsk ressursfil:

```
# -- mainMenu --
mainMenu.title=Hovedmeny
mainMenu.heading=Velkommen, {0}!
mainMenu.login=Logg inn
mainMenu.message=Du er n&aring; logget inn. Vennligst foreta et valg fra hovedmenyen.
```

Nøklene er på venstre siden av likhetstegnene, og de norske oversatte tekstene til høyre.

Vi ønsker å benytte ressursfiler for oversettelse av all språkavhengig tekst i publikumsmodulen, bortsett fra tekst som kan endres av systemets administratorer eller andre via systemets brukergrensesnitt.

6.5.2 Lokalisering ved hjelp av databasetabeller

Universell Utforming systemet har også behov for oversettelser av tekst som er registrert via systemets brukergrensesnitt. Denne teksten lagres i utgangspunktet i MySQL databasen, og der bør også oversettelsene lagres. Prinsippet er at det lagres en oversatt tekst per definert nøkkel og språk i systemet.

Nøkkel	Språk	Oversettelse
language.inlanguage	No	På Norsk
language.inlanguage	En	In English
language.inlanguage	Fr	En Français

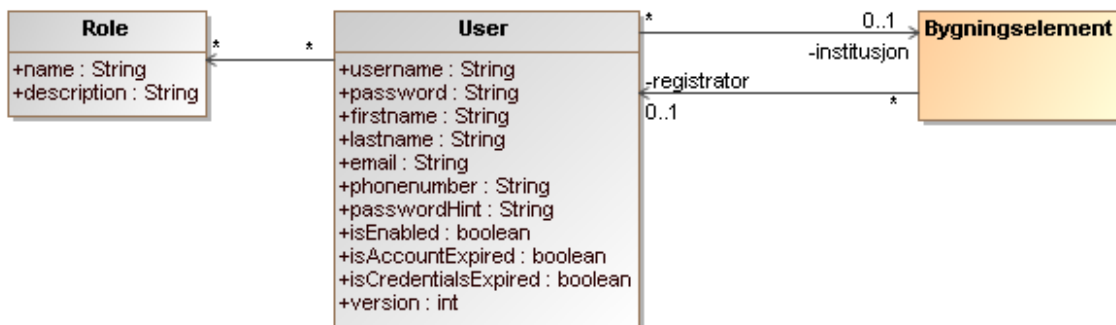
Vi lager et enkelt brukergrensesnitt for å identifisere ikke oversatte uttrykk, og viser to språk ved siden av hverandre. Ett av språkene kan brukes som referanse. Det andre kan redigeres.

6.5.3 Oversetting til et nytt språk

Arbeidsprosessen med å oversette Bygg for alle til et nytt språk blir dermed todelt:

1. Man tar kopi av en av de eksisterende ressursfilene, oversetter teksten til høyre for likhetstegnene til det nye språket, mens man passer på å ikke endre nøklene på venstre siden av likhetstegnene. Den nye ressursfilen kopieres deretter inn i systemet med riktig navn i forhold til språket det ble oversatt til.
2. Administrator legger til det nye språket i oversikten over hvilke språk systemet støtter. Oversetter starter brukergrensesnittet for oversetting, henter oversikten over ikke oversatte uttrykk, og oversetter uttrykkene i sitt eget tempo. Deretter gis språket status som offentlig tilgjengelig.

6.6 Brukere og roller



Figur 9 Bruker/rolle

6.6.1 Roller

Hver bruker har et sett av roller. Rollene som finnes i systemet er forhåndsdefinert og kan ikke endres av administrator. De tilgjengelige rollene er:

- Systemeier
- Institusjonsforvalter
- Eiendomsforvalter
- Registrator
- Kvalitetssikre
- Offline registrator

Offline registratorrollen brukes bare for lokale installasjoner for registrering, på den sentrale installasjonen (www.byggforalle.no) er ikke Offline registratorrollen i bruk.

6.6.2 Brukere

Brukeres passord er lagret kryptert med SHA algoritmen.

7 Autentisering og innlogging

BFA benytter Spring Acegi Security som rammeverk for autentisering og tilgangsstyring. Innloggingsflyten er utvidet med totrinnsverifisering (2FA) via SMS.

7.1 Innloggingsflyt

Innlogging skjer i to trinn:

Verifiseringskode: Brukeren navigerer til `/logintwofactor.html`¹ og oppgir brukernavn samt registrert mobilnummer. Systemet genererer en 6-tegns alfanumerisk kode og sender den som SMS via AWS SNS. Koden har en konfigurerbar gyldighetsperiode (standard satt via `twoFactor.minutes.validity`).

Innlogging: Brukeren navigerer til `/login.html` og oppgir brukernavn, passord og verifiseringskoden mottatt på SMS. Alle tre må stemme for at innlogging skal godkjennes.

7.2 Totrinnsautentisering (2FA)

For 2FA gjelder følgende:

- Koden er 6 tegn lang og består av tall og bokstaver (store og små), men utelater tegn som lett forveksles: I, l, O og 0.
- Koden genereres med `SecureRandom` for kryptografisk sikkerhet.
- Koden lagres i databasen sammen med et tidsstempel og et flagg som markerer om koden allerede er brukt.
- En kode kan kun brukes én gang.
- Ved vellykket 2FA-innlogging markeres mobilnummeret som verifisert.
- Avsender-ID for SMS-meldingen er satt til BFA.

Dersom brukeren ikke har registrert mobilnummer, eller nummeret ikke stemmer, vil systemet avvise forespørselen om ny kode.

7.3 Passord

7.3.1 Krav til Passord

Passord må oppfylle følgende minimumskrav:

- Minst 8 tegn
- Minst én stor bokstav
- Minst én liten bokstav
- Minst ett siffer

¹ Dette og lignende lenker tilsvarer f.eks. <https://forvaltning.byggforalle.no/uu/logintwofactor.html>

7.3.2 Hashing og lagring

Passord lagres aldri i klartekst. De hashes med SHA-algoritmen og et per-bruker tilfeldig generert salt (8 tegn). Salt lagres som et eget felt på brukerobjektet.

7.3.3 Passordutløp

Passord utløper etter et konfigurerbart antall dager. Dersom passordet er utløpt bes brukeren om å sette nytt passord via «Glemt passord?»-funksjonen. Brukere varsles i forkant av utløp: Antall gjenværende dager og utløpsdato vises i brukergrensesnittet.

7.3.4 Endre passord

Innloggede brukere kan endre eget passord via /editPassword.html. Det nye passordet må oppfylle passordkravene og kan ikke være identisk med det eksisterende passordet.

7.4 Glemt passord / tilbakestilling

Dersom en bruker har glemt passordet, kan de be om en tilbakestillingskode via /forgotPassword.html:

1. Brukeren oppgir brukernavnet sitt.
2. Systemet genererer en 6-tegns tilfeldig kode (passwordCode) og lagrer den med et tidsstempel (passwordCodeTimestamp) på brukerobjektet.
3. Koden sendes til brukerens registrerte e-postadresse via en e-postmal (passwordCode.vm).
4. Brukeren bruker koden på /resetPassword.html for å sette nytt passord.

E-posten sendes fra byggforalle@statsbygg.no og besvares ikke.

7.5 Ny bruker og verifikasjon av mobilnummer

Etter at nye bruker opprettes må bruker første gang gjennom en verifikasjon av at mobilnummer er riktig. Dette gjelder også for brukere som ikke er definert med mobilnummer eller skal bytte mobilnummer. Brukeroversikt viser om mobilnummer er verifisert.

Mobilnummer er i dag et obligatorisk felt, men det har ikke alltid vært sånn. Derfor finnes det brukere uten mobilnummer eller med fast telefonnummer som mobilnummer

7.6 Brukernavn og e-postadresse

Merk at det er et 1-til-1 forhold mellom «brukernavn» og e-postadresse. Det betyr at e-postadresse og brukernavn er tett knyttet sammen som id. Dette betyr at hvis noen ønsker flere brukere (med ulik rolle) må disse ha forskjellige e-postadresser. Mens flere brukere kan ha samme mobilnummer, fornavn og etternavn uten at det gir problemer.