

Recruit

En plattform for en likestilt rekrutteringsprosess

BACHELOR 2022

Forfattere

Nima Abdollahi

Glaysa Fernandez

Sheima Al-Shamarti

Ali Reza



Institutt for Informasjonsteknologi
 Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
 Besøksadresse: Holbergs plass, Oslo

PROSJEKT NR.
13
TILGJENGELIGHET
Åpen

Telefon: 22 45 32 00

BACHELORPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
ARecruit: Anonym Rekrutteringsplattform	04.01.2022
	ANTALL SIDER / BILAG
	110
PROSJEKTDELTAKERE	INTERN VEILEDER
Nima Abdollahi (S341890) Glaysa Fernandez (S344047) Sheima Al-Shamarti (S338853) Ali Reza (S341823)	Roza Abolghasemi

OPPDAGSGIVER	KONTAKTPERSON
WAYS AS	Bjørn-Ivar Skuggen

SAMMENDRAG
I samarbeid med WAYS AS har gruppen utviklet en webapplikasjon som skal gjøre det mulig for en kandidat å søke til en stilling og samtidig være anonym frem til kandidaten blir tilkalt til intervju. Denne webapplikasjonen er en løsning for fordommer mot alder, kjønn, etnisitet, osv. Den skal bidra til å gjøre rekrutteringsprosessene til bedrifter mer inkluderende.

3 STIKKORD
Anonymisert Rekrutteringsportal
Webapplikasjon
Inkluderende Jobsøking

Forord

Mangfold inkluderer ikke bare nasjonalitet, etnisitet eller kjønn, men også alle de tingene som gjør oss unike: vår personlighet, bakgrunn, ferdigheter, interesser og perspektiver. På grunn av våre forskjeller kan mange av oss ha bevisste og ubevisste fordommer mot hverandre. Dette skjer i mange forskjellige situasjoner. En av disse situasjonene er under rekrutteringsprosesser.

I løpet av dette studieåret (2021/22) har vi hatt privilegiet til å jobbe med en løsning for å utelukke fordømmelser i rekrutteringsprosesser. Denne løsningen er et bachelorprosjekt for to treårige bachelorstudier innenfor dataingeniør og informasjonsteknologi ved fakultetet teknologi, kunst og design ved OsloMet universitet i Oslo.

Vår takknemlighet går til WAYS AS for å ha gitt oss denne muligheten å jobbe med en samfunnsnyttig problemstilling. Ikke minst ønsker vi å takke våre veiledere fra WAYS: Bjørn-Ivar Skuggen og Simen Burud for støtten de har gitt gruppen gjennom utviklingsprosessen med innspill og faglig bistand. Tilslutt ønsker vi å takke vår internveileder Roza Abolghasemi for hjelpen vi har fått under rapporten.

Oslo - 04.01.2022

Nima Abdollahi
Glaysa Fernandez
Sheima Al-Shamarti
Ali Reza

Rapportstruktur

Denne rapporten består av fem kapitler. Kapittel 1 er en presentasjon av oppgaven, gruppemedlemmene og veilederne. Kapittel 2 er en detaljert gjennomgang av prosessen i forskjellige faser. Kapittel 3 og 4 består av produktdokumentasjon og testdokumentasjon av løsningen. Kapittel 5 er brukermanual og inneholder en detaljert framgang for utføring av de forskjellige oppgavene på webapplikasjonen.

Rapporten begynner med en felles innholdsfortegnelse som omdanner alle kapitlene detaljert. I begynnelsen av hvert kapittel finnes det en kort innholdsliste av temaene. Dette er gjort for å gi en god orientering. Fotnoter er brukt gjennom rapporten for å gi en kort forklaring på begreper og forkortelser. De er også lenket med ordlisten som kommer i slutten av dokumentet. I tillegg finnes det referanser som er lenket med bibliografi delen med et tall (f.eks. [\[1\]](#)).

Innholdsfortegnelse

BACHELORPROSJEKT	2
Forord	3
Rapportstruktur.....	4
Innholdsfortegnelse	5
Presentasjon	10
1.1 Introduksjon.....	11
1.1.1 Presentasjon av gruppen	11
1.1.2 Presentasjon av oppdragsgiver	13
1.1.3 Kontaktpersoner fra WAYS AS.....	13
1.1.4 Intern veileder fra OsloMet.....	14
1.2 Presentasjon av prosjektet	14
1.3 Løsning	15
Prosessdokumentasjon	16
2.1 Innledning	17
2.2 Oppstart	17
2.2.1 Fremdriftsplan	18
2.2.2 Planlegging og kommunikasjonsverktøy	19
2.2.3 Modellering av database	21
2.3 Designfase	23
2.3.1 Brukergrensesnitt.....	23
2.3.2 Brukeropplevelse	23
2.3.3 Ikonene og logo	23

2.4 Utviklingsfase	25
2.4.1 Sprinter	25
Sprint 1.....	25
Sprint 2.....	27
Sprint 3.....	28
Sprint 4.....	30
Sprint 5.....	30
Sprint 6.....	31
2.5 Endringer og utfordringer	32
2.5.1 Mappe Restrukturering.....	32
2.5.2 Designendringer	32
2.5.3 Rammeverk Oppgradering	33
2.6 Konklusjon av prosessen.....	34
2.6.1 Avvik i forhold til fremdriftsplan	34
2.6.2 Konklusjon og refleksjoner.....	34
Produktdokumentasjon	35
3.1 Introduksjon til ARecruit - Anonymisert rekrutteringssystem	36
3.2 Produktspesifikasjoner.....	36
3.2.1 Kravspesifikasjon	36
3.2.1.1 Ikke-funksjonelle krav.....	37
3.2.1.2 Funksjonelle krav	37
3.2.2 Aktivitetsdiagram	39
3.3 Frontend	41
3.3.1 Teknologier og verktøy	42
3.3.1.1 Angular.....	42
3.3.1.2 Bootstrap	42
3.3.2 Hovedsider	43
3.3.2.1 Innloggingsside	43
3.3.2.2 Passord reset siden	43
3.3.2.3 Stilling dashboard for admin.....	45
3.3.2.4 kandidat dashboard for admin.....	45

3.3.2.5 PDF-editor for admin	46
3.3.2.6 PDF-viser for admin	47
3.3.2.7 Søknadsportal.....	48
3.3.3 Komponenter	49
3.3.3.1 Navigasjonskomponent.....	51
3.3.3.2 Kandidat sammendrag komponenter	52
3.3.3.3 Alert komponent	53
3.3.3.4 Filoplastningskomponent	54
3.3.3.5 Kort komponenter	54
3.3.4 Modeller	55
3.3.5 Route guards og resolvers	55
3.3.6 Services	57
3.3.6.1 HTTP Services.....	57
3.3.6.2 Authentication Services	57
3.3.6.3 Inform Candidate Services	58
3.3.6.4 Alert Services	58
3.3.6.5 Password Services.....	59
3.4 Backend	59
3.4.1 Teknologier og verktøy	60
3.4.1.1 C# ASP.NET Core.....	60
3.4.1.2 Entity Framework.....	60
3.4.2 Web Layer	60
3.4.2.1 Controllers.....	60
3.4.2.2 Dtos (Data Transfer Objects).....	62
3.4.2.3 Contracts.....	62
3.4.3 Data Access Layer (DAL)	63
3.4.3.1 Models	63
3.4.3.2 Repositories	64
3.5 Sikkerhet.....	64
3.5.1 Datavalidering	65
3.5.2 Autentisering og autorisering.....	65

3.6 Filstruktur.....	66
3.6.1 Backend.....	66
3.6.2 Frontend	69
3.7 Konklusjon	71
Testdokumentasjon	72
4.1 Introduksjon.....	73
4.1.1 Testplan.....	73
4.1.2 Avvik i forhold til testplanen	74
4.2 Enhetstesting	75
4.2.1 Testobjekter.....	75
4.2.2 Testmetodikk og testverktøy	76
4.2.3 Konklusjon av enhetstesting.....	78
4.2.3.1 Resultater	78
4.2.3.2 Konklusjon	79
4.3 Integrasjonstesting	79
4.3.1 Testobjekter.....	79
4.3.2 Testmetodikk og testverktøy	80
4.3.3 Konklusjon av integrasjonstesting.....	81
4.3.3.1 Resultater	81
4.3.3.2 Utfordringer og læringsutbytte	82
4.3.3.3 Konklusjon	83
4.4 Test oppsummering	84
Brukermanual	85
5.1 Logg inn side.....	86
5.1.1 Innlogging for administrasjon brukere	86
5.1.2 Tilbakestilling av passord	87
5.2 Inspektør.....	88
5.2.1 Anonymisering av kandidater	88
5.2.2 Kontakt kandidat	92

5.3 Beslutningstaker.....	93
5.3.1 Vurdering av kandidater.....	93
5.3.2 Legg til nye stillinger	96
5.4 Jobbsøking for kandidater.....	97
5.5 Andre komponenter	102
5.5.1 Navigasjonsmenyen.....	102
5.5.2 Filtrere kandidater	103
Ordliste	103
Bibliografi.....	107

Kapittel 1

Presentasjon

Innhold

- 1.1 Introduksjon**
- 1.2 Presentasjon av prosjektet**
- 1.3 Løsning**

1.1 Introduksjon

I samarbeid med WAYS AS har gruppen utviklet en webapplikasjon som skal gjøre det mulig for en kandidat å søke til en stilling og samtidig være anonym frem til kandidaten blir tilkalt til intervju. Denne webapplikasjonen er en løsning for fordommer mot alder, kjønn, etnisitet, osv. Den skal bidra til å gjøre rekrutteringsprosessene til bedrifter mer inkluderende.

Dette kapitlet inneholder en introduksjon til bachelorprosjektet. De temaene som befinner seg her er en kort beskrivelse av problemstillingen og løsningen. I tillegg skal det introdusere oppdragsgiveren, veilederne og gruppemedlemmene.

1.1.1 Presentasjon av gruppen

Gruppen består av fire teknologi studenter fra linjene dataingeniør og informasjonsteknologi på OsloMet. Vi har tidligere jobbet sammen i flere andre mindre gruppeprosjekter på skolen, og bestemte oss for å jobbe sammen i dette prosjektet.



Nima Abdollahi

Nima er en dataingeniør student som har alltid hatt interesse for teknologi. Han har valgt denne studien for å bidra med en enklere hverdag i den digitale verden. Utenfor studien er han en aktiv person og er glad i planter.

E-post: nimabewrani98@gmail.com

Telefonnr: +47 400 56 425

**Glaysa Fernandez**

Glaysa sin interesse for programmering startet i ung alder..

Informasjonsteknologi er studieretningen hun har valgt å ta.

Hun engasjerer seg med både backend og frontend. Glaysa er en konkurransedyktig person. Personer som har bedre kompetanse enn hun, oppmuntrer hun å sette høye ambisjoner og motiverer hun å forbedre kompetansene sine.

E-post: glaysa.df@gmail.com

Telefonnr: +47 939 88 118

**Sheima Al-Shamarti**

Sheima går på studieretningen dataingeniør, og har alltid hatt interesse for teknologi. Hun har også tidligere jobbet som studentassistent i et matematikk emnet på OsloMet.

Som person er hun sosial, aktiv og liker å tilegne seg nye kunnskaper.

E-post: sheima.ali@hotmail.com

Telefonnr: +47 400 61 746

**Ali Reza**

Ali er dataingeniørstudent ved OsloMet og jobber som studentassistent i matematikk - 3-terminsordning. Han er en aktiv, sosial, og lærevillig. Ali har planer med å ta et master innenfor IT-sikkerhet etter endt studie.

E-post: h.alireza96@gmail.com

Telefonnr: +47 409 48 245

1.1.2 Presentasjon av oppdragsgiver

WAYS AS er et digitalt utviklingsbyrå i Oslo som utvikler mobile applikasjoner, større bedriftsnettsider og plattformløsninger. WAYS AS har stilt opp med veiledning i det tekniske aspektet av prosjektet og designet brukergrensesnittet.

Kontaktinformasjon til WAYS AS:

Adresse: Øvre slottsgate 11, 0157 Oslo

E-post: hei@ways.no

Telefonnr: 475 17 000



1.1.3 Kontaktpersoner fra WAYS AS

Ekstern veileder

Bjørn-Ivar Skuggen jobber som utvikler hos WAYS AS og var vår ekstern veileder i dette prosjektet og hovedkontaktperson. Gruppen har hatt jevnlige møter med Bjørn annenhver uke. Han har hovedsakelig hjulpet oss med design endringer, endringer i kravspesifikasjonen og funksjonaliteter.



E-post: bjorn.ivar.skuggen@ways.no

Teknisk veileder

Simen Burud jobber hos WAYS AS som utvikler. Han var teknisk veileder for gruppen gjennom prosjekt løpet. Vi fikk hjelp med diverse problemer og spørsmål fra han som brukergrensesnitt, database design, testing og mer.



E-post: simen@ways.no

1.1.4 Intern veileder fra OsloMet

Roza Abolghazemi har vært gruppens interne veileder, og hjulpet oss med det som omhandler dokumentasjon gjennom prosjektet. Vi hadde fått gode tips i starten om hvordan starte opp et bachelorprosjekt, ting vi burde tenke på mens vi jobber, og har siden da holdt kontakten sammen.



E-post: roza.abolghasemi@oslomet.no

Telefonnr: +47 459 16 836

1.2 Presentasjon av prosjektet

Som det er nevnt i forprosjektrapporten, så mener Ways at mangfold er en rød tråd i all vår virke. De ønsker å benytte teknologi til å forbedre og gjøre arbeidsplassen mer inkluderende. Rekrutteringsprosesser er komplekse og administrerende direktør av Ways ønsker seg å fortsette å være nøytral. “*Forskning viser at vi i mange tilfeller gjør feilansettelser og diskriminerer i slike prosesser til tross for at vi i beste mening ønsker å forholde oss nøytrale*” Sier Knut Michael, CEO i WAYS AS. [1]

1.3 Løsning

Ved å lansere en anonymisert rekrutteringsportal, hindrer vi fordommene vi tar med oss inn i en rekrutteringsprosess. Grunnen er at HR får ikke vite andre informasjon annet enn ferdigheter, utdanning og andre relevante informasjon om kandidaten for stillingen. Etter oppdragsgiverens ønske om en slik løsning har gruppen jobbet med å utvikle en webløsning som skal tilfredsstille kravene til oppdragsgiver.

Løsningen består av tre selvstendige systemer. De er følgende:

1. Interaktivt grensesnitt som er utviklet for browser. Den sørger for en god bruker interaksjon med API-et.
2. API¹ laget som sørger for tilgang til data og interne system funksjonaliteter.
3. Database² systemet som er ansvarlig for lagring og lesing av data.

¹ [API](#) er en forkortelse for Application Programming Interface som er en mellommann som kobler flere deler av applikasjoner eller systemer sammen.

² [Database](#) er en organisert kolleksjon av strukturert data og informasjon.

Kapittel 2

Prosessdokumentasjon

Innhold

- 2.1 Innledning**
- 2.2 Oppstart**
- 2.3 Designfase**
- 2.4 Utviklingsfase**
- 2.5 Endringer og utfordringer**
- 2.6 Konklusjon av prosessen**

2.1 Innledning

I dette kapitlet av prosjektrapporten har det blitt dokumentert utviklingsprosessen for prosjektet fra oppstart til slutten. Her går vi gjennom vår arbeidsmetode, utfordringer, løsningsmetoder og beslutninger. Dette kapitlet baserer seg på en dagbok som gruppemedlemmene har ført sammen hver dag for å holde styr på prosessen.

2.2 Oppstart

Gruppen begynte arbeidet med bachelorprosjektet fra 4.januar 2022. På den dagen holdt gruppen en orienteringsmøte for å planlegge arbeidet for første og andre uke i januar. I dette møtet kom vi til enighet om å holde møte med intern veileder og vår oppdragsgiver for å få en mer oppklaring om prosjektet hos ekstern veileder og planlegge møter med intern veileder.

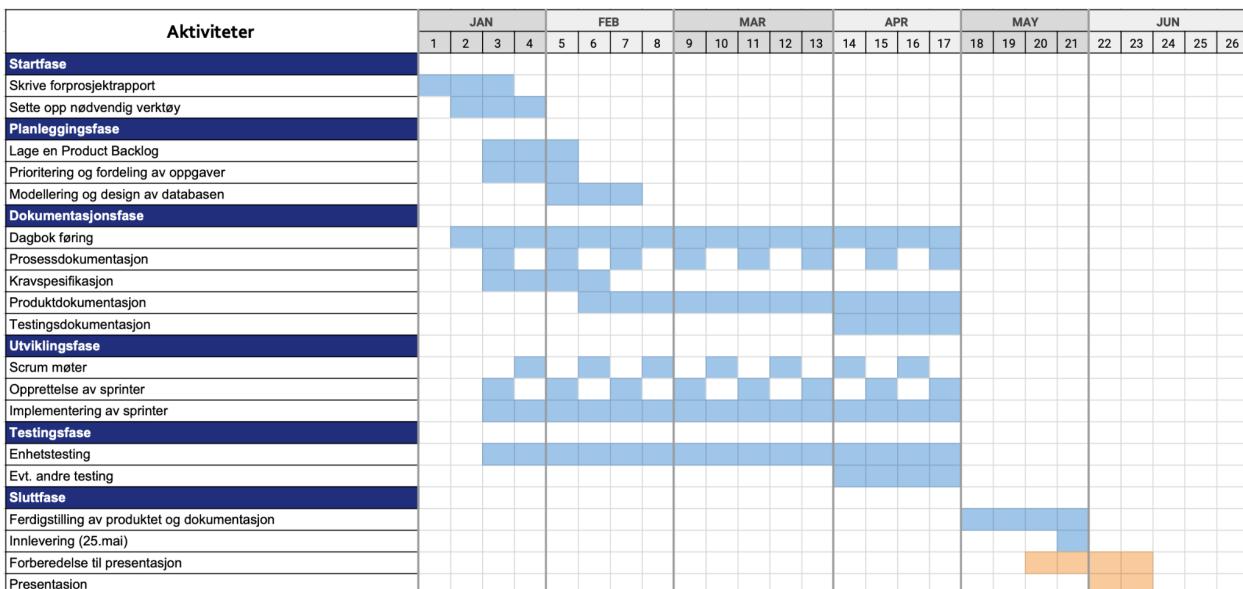
Den første uken av januar brukte vi på grunnleggende planlegging av prosjektet og skrev ferdig første utskrift av forprosjektrapporten. Vi hadde også møte med ekstern veileder hvor vi fikk oppklart spørsmål vi hadde om oppgaven og diskuterte andre aspekter ved prosjektet som blant annet arbeidsmetodikk vi tenkte å bruke og verktøy.

I andre uken av januar fikk vi møte vår intern veileder, Roza Abolghasemi, og ble kjent med henne. Vi hadde blant annet spørsmål om hvordan hun kommer til å være til hjelp for oss i gruppen og avtalte hvordan våre fremtidige møter vil være. Vi ble blant annet enige om fleksible møter istendefor faste ukentlig møter. Vi kom fram til at vår intern veileder hjelper oss med dokumentasjon, mens vår eksterne veileder stiller til hjelp med det tekniske. Vi kommer til å sende utkast av dokumentasjon via mail som vi kommer til å få tilbakemeldinger av fra Roza. Vi har også brukt resten av andre uke

på å starte prosjektet, opprettet sprinter og delte oppgaver mellom gruppemedlemmer.

2.2.1 Fremdriftsplan

I andre uken av januar lagde gruppen en fremdriftsplan for arbeidsprosessen. Denne planen ble laget ut fra at vi hadde et ferdiglaget design og at vi jobbet i en smidig metodikk. Planen baserte seg på rammebetingelsene gruppen har hatt for prosjektet, blant annet at prosessen skulle følge en smidig arbeidsmetodikk, hvor vi hadde sprint møter to ganger i måneden. En annen rammebetingelse er at gruppen har fått en ferdig design av produktet som skulle implementeres. Vi tok utgangspunkt i at vi blir ferdig med prosjektet en måned før innleveringen, dette slik at vi har resten av tiden til å se over og ferdigstille. På den måten er det enkelt å gjøre endringer uten at vi føler på tidspress.



Figur 2.2.1: Fremdriftsplan

2.2.2 Planlegging og kommunikasjonsverktøy

Zoom er et kommunikasjonsverktøy som brukes for videokonferanser.

Vi brukte zoom hovedsakelig for å holde møte med vår intern veileder.

[2]



I likhet med zoom er også Google Meet et kommunikasjonsverktøy og brukes til videokonferanser og nettmøter. Google Meet er et brukervennlig verktøy og tilbyr blant annet skjermdeling og lar oss planlegge og delta i forhåndsbestemte møter. Vi har brukt Google Meet for gruppearbeid og for scrum møter med arbeidsgiveren. [3]



Git er en åpen kildekode distribuert versjonskontrollsysten der utviklere har en versjon av koden i sin egen lokal maskin og en annen versjon som blir delt på en server. Ved hjelp av innebygde funksjoner som git har, gjør det lett for utviklere å samarbeide på et prosjekt samtidig. [4]



Jira er et smidig prosjektstyring verktøy utviklet av Atlassian. Jira brukes for sporingssystem for smidig arbeidsmetodikk, [5]. Gruppen brukte dette verktøyet for å sette opp sprints og backloger for disse. Også fordelte arbeidet innenfor gruppen ved hjelp av Jira.



Figma er et vektor grafisk redigering og prototype verktøy som brukes for å designe mobilapper og webapplikasjoner, [6]. Vi hadde fått ferdig designet prototype av applikasjonen på figma og derfor trengte vi ikke selv å designe den. Men vi brukte verktøyet som en referanse for utvikling av frontend, for å designe logoen og designe sluttdokumenteten vår.



Google Sheets er et Google Workspace dataprogram og er basert på regneark der vi kan utføre ulike matematiske beregninger og analyser, [7]. Google Sheets har vært til stor nytte for oss vi brukte programmet for å sette opp vår fremdriftsplan. På grunn av programmets funksjoner og effekter, satt vi opp en god plan som var lett å følge. I tillegg brukte vi Sheets for integrasjonstesting der vi skrev opp alle testcaser på et Google Sheets regneark.



Diagrams.net eller draw.io er en åpenkildekode programvare som er utviklet i html og javaScript, [8]. Gruppen brukte programvaren for å designe vår datamodellering og for å sette opp aktivitetsdiagrammer.



Slack er en kommunikasjonsplattform og er godt egnet for samarbeid i grupper, [9]. Slack lar oss opprette kanaler for gruppen og støtter blant annet fildeling og bildedeling. Vi brukte slack mest for å opprettholde kontakt med vår veileder fra WAYS AS og planlegge scrum møter med.

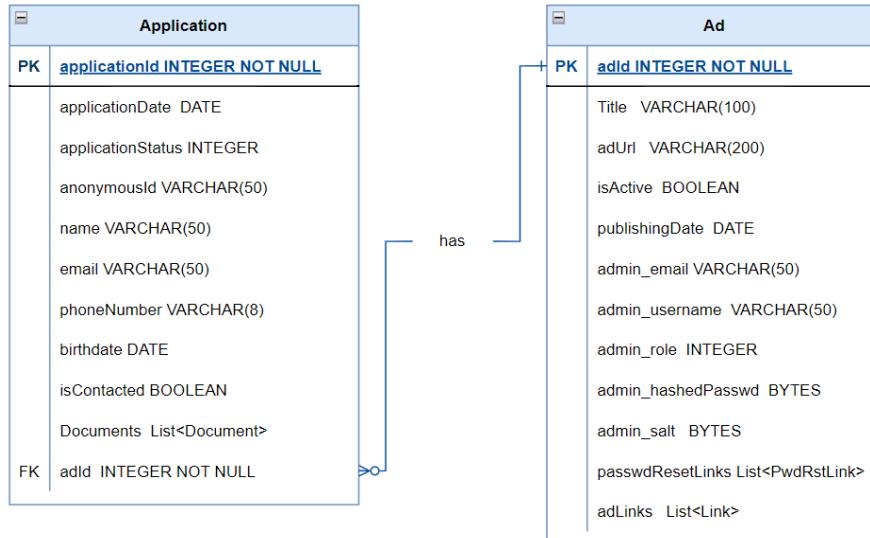


2.2.3 Modellering av database

En viktig del av planleggingsfasen besto av database design og struktur av datalagring, for data i seg selv har ingen nytte før den er strukturert. Etter strukturering og modellering av data kan de gi mening og informasjon.

Databasemodelleringen skjedde gjennom flere steg, hvor i hvert steg prøvde vi å forbedre databasen slik at vi unngikk redundante og kompliserte rader. Denne prosessen er kalt for *Database Normalisering*.

På første steg forberedte vi en enkelt og overfladisk design av databasen med de grunnleggende tabeller som systemet krevde. På slutten av dette steget hadde vi laget en database som var på første normalform. I denne normalformen besto tabellene av kolonner som inneholdt flere rader. Dette kan pekes på i Figur 2.2.3-A, hvor tabellen *Application* har en liste av dokumenter og dette resultere i flere rader i denne kolonnen.



Figur 2.2.3-A: Første skisse av diagram på databasen

Etter første steget som ga et design på første normalform, begynte vi med andre steget som var å hente ut de kolonnene som besto av flere rader. For disse kolonnene lagde vi tabeller og brukte deres primærnøkkel som fremmednøkkel i stedet. Dette kan vi se i *Figur 2.2.3-B*, hvor *Application* tabellen har *CVId* og *LetterId* kolonnene som er fremmednøkler fra *Document* tabellen. Lignende eksempler kan vi se i relasjonene mellom *Ad* og *Link*, og *Admin* og *PasswdResetLink*. Til slutt ble resultatet av dette steget til en ferdig design av databasen som er på tredje normalform. Og vi har brukt denne databasen designen for å løse oppgaven. [10]

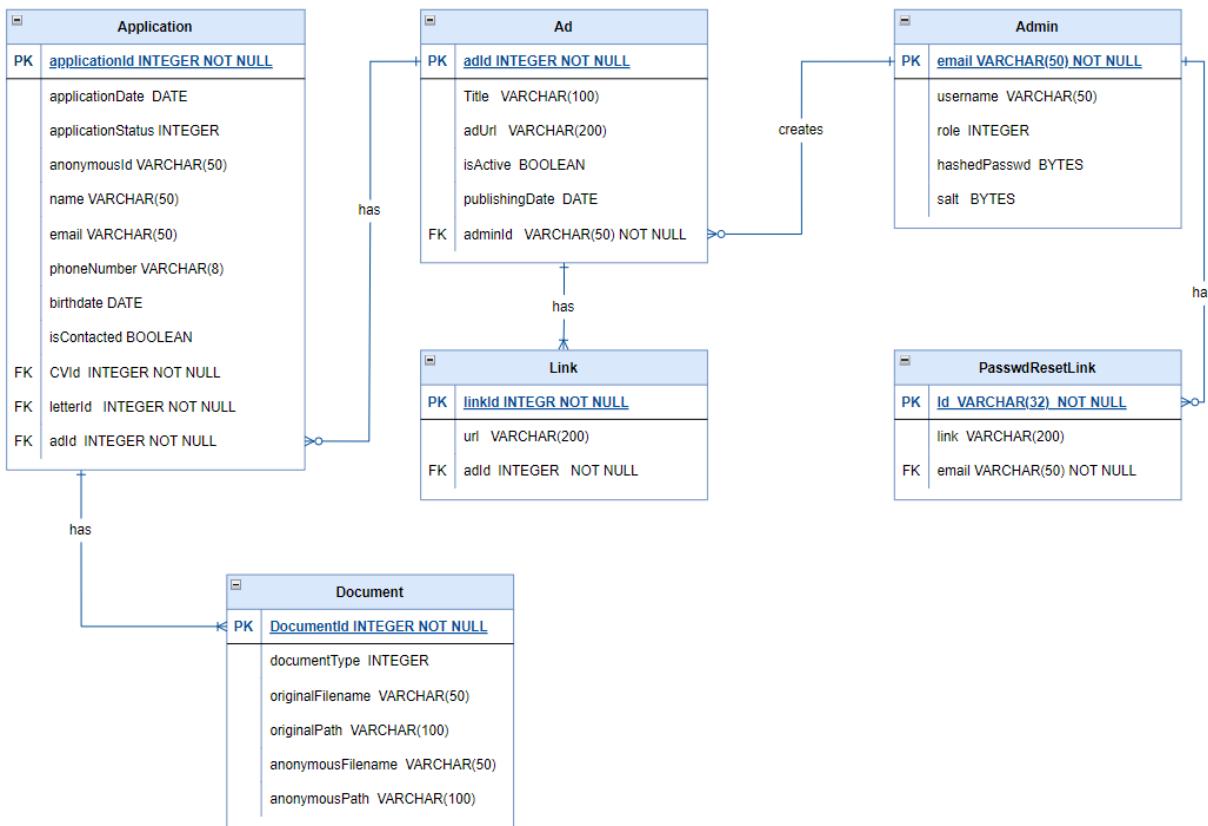


Figure 2.2.3-B: ER-modellen på tredje normalform (3.NF)

2.3 Designfase

2.3.1 Brukergrensesnitt

Før utviklingen av applikasjonen ble satt i gang, fikk gruppen et utkast av applikasjonens brukergrensesnitt fra oppdragsgiver i desember 2021. Designet ble ferdigstilt i første uken av januar 2022. Gruppen hadde tilgang til designet gjennom *Figma* som gjorde det lettere for oss å se de nødvendige stilene til hver element som trengs for brukergrensesnittet. Vi stod fritt til å endre designet, men gruppen valgte å holde fast ved det gitte designet og bestemte oss å gjøre endringer når det kun var nødvendig. Under utviklingen, ble det gjort noen designendringer både av oss og oppdragsgiver. Disse endringene er nevnt under [*Designendringer*](#) seksjon.

2.3.2 Brukeropplevelse

Opplevelsen av brukerne når de interagere med brukergrensesnittet er viktig. Med vår kunnskap fra *Webutvikling* og *inkluderende design* og *Menneske maskin interaksjon* har det blitt en konvensjon å gjøre applikasjoner med dagens moderne teknologi brukervennlige.

2.3.3 Ikonene og logo

Ikonene er hentet både fra designet vi fikk fra oppdragsgiver og et ikon-bibliotek: *Bootstrap Icons*. Logoen er designet av gruppen og er inspirert fra oppdragsgiverens logo design. Vi hadde et par utkast av logo designet og endte opp med to logoer.



Figur 2.3.3-A:
Valgt favicon design



Figur 2.3.3-B: Utkast av favicon design

Figur 2.3.3-A er favicon designen som gruppen valgt å bruke. I tillegg til faviconen, designet gruppen en større logo som representerer plattformen på et tydeligere måte som er på Figur 2.3.3-C.



Figur 2.3.3-C: Plattformens logo

2.4 Utviklingsfase

2.4.1 Sprinter

Hver sprint³ varte i to uker. Sprintene begynte med planlegging av arbeidet og skriving av backlog⁴ som måtte bli fullført innenfor den gitte tidsrammen. Vi omtalte ukene i hver sprint som første uke og andre uke.

Sprint 1

Hva var planen for sprinten?

Vi hadde fått et ferdig design av oppdragsgiver som kun trengte å bli implementert, derfor var planen da for denne sprinten å få ferdigstilt designen av prosjektet slik at vi kunne raskt gå videre med andre deler av prosjektet som er mer tidskrevende.

For denne sprinten hadde vi som mål å fullføre følgende brukerkrav på frontend:

- Som kandidat vil jeg kunne fylle opp mine personlige opplysninger, for å kunne registrere meg som en kandidat på stillingen.
- Som kandidat vil jeg kunne laste opp mine dokumenter (cv/søknadsbrev) som er relevant for stillingen jeg søker på.
- Som kandidat ønsker jeg mulighet til å skrive søknadstekst for å kunne vise min interesse for stillingen.
- Som kandidat ønsker jeg å se hvor jeg er i prosessen for å kunne holde oversikt over søknaden.
- Som inspektør ønsker jeg en tavle for å se aktive og arkiverte stillinger, slik at jeg kan se om det har kommet søknader og å holde oversikt på stillingene.

³ [Sprint](#) er en iterasjon av utviklingsprosess i softwareutvikling, det varer i 1-2 uker som vanlig.[\[11\]](#)

⁴ [Sprint backlog](#) er en liste av planen som skal utføres i sprinten.[\[11\]](#)

- Som inspektør ønsker jeg en oversikt over kandidater som ikke er anonymisert enda, for at jeg kan starte anonymiseringsprosessen og å sende dem videre til godkjenning.
- Som inspektør ønsker jeg statusoversikt over kandidater som er godkjent, venter på godkjenning eller ikke godkjent, slik at jeg kan kontakte dem for invitasjon eller avslag.
- Som inspektør ønsker jeg en sensureringsverktøy, slik at jeg kan sensurere personopplysninger og eventuell andre opplysninger.
- Som beslutningstaker ønsker jeg oversikt over kandidater som er anonymisert, slik at jeg kan gå gjennom søknadene deres.
- Som beslutningstaker ønsker jeg mulighet til å legge til nye stillingsannonser, slik at kandidater kan søke på stillingen.
- Som beslutningstaker og inspektør ønsker vi en navigasjonsliste, slik det blir mer brukervennlig å navigere i applikasjonen.
- Som beslutningstaker og inspektør krever vi en logg inn side, for å kunne logge oss inn på applikasjonen, slik at vi kan utføre våre oppgaver.

Hva fullførte vi?

Vi fullførte mange av kravspesifikasjonene til sprinten, men det var noen få krav som ikke ble utført i denne sprinten som ble tatt med videre til sprint 2, og dermed prioriterte vi disse først før startet på nye oppgaver.

Disse brukerhistoriene var følgende:

1. Som kandidat vil jeg kunne laste opp mine dokumenter (cv/søknadsbrev) som er relevant for stillingen jeg søker på.
2. Som inspektør ønsker jeg en sensureringsverktøy, slik at jeg kan sensurere personopplysninger og evt. andre opplysninger.

3. Som beslutningstakker og inspektør krever vi en logg inn side, for å kunne logge oss inn på applikasjonen, slik at vi kan utføre våre oppgaver.

Hvilke utfordringer oppstod?

Last opp pdf fil hadde noen komplikasjoner som gjorde at den ikke helt fungerte som den skulle ved at den lagret siste fil som ble lastet opp, uavhengig av hvilken side opplastingen oppstod (CV eller søknadsbrev). Dette ble løst ved at flere i gruppa jobbet med problemet og prøvde ut forskjellige løsninger til vi fant en som fungerte. I tillegg hadde vi utfordringer med implementasjon av sensureringsverktøyet. Dette kravet ble flyttet til sprint 2.

Sprint 2

Hva var planen for sprinten?

I denne sprinten var planen først å bli ferdig med de resterende oppgavene fra sprint 1 i første uke av sprint 2, deretter gikk vi videre til modellering av databasen som er en del av planleggingsfasen på vår fremdriftsplan.

De resterende brukerkarav er følgende:

1. Som kandidat vil jeg kunne laste opp mine dokumenter (cv/søknadsbrev) som er relevant for stillingen jeg søker på.
2. Som inspektør ønsker jeg en sensureringsverktøy, slik at jeg kan sensurere personopplysninger og evt. andre opplysninger.
3. Som beslutningstakker og inspektør krever vi en logg inn side, for å kunne logge oss inn på applikasjonen, slik at vi kan utføre våre oppgaver.

Hva fullførte vi?

Vi fullførte de forsinkende oppgavene fra sprint 1 i første uken av denne sprinten. Den påfølgende uken, diskuterte gruppen databasedesignen og strukturen. For å enkelt

identifisere listen over tabeller i databasen og deres relasjoner med hverandre, har gruppen laget diagrammer som representerer disse.

Hvilke utfordringer oppstod?

Det har ikke vært noen konkrete utfordringer, men vi har hatt noen spørsmål angående datalagring og det ble tatt opp med ekstern veileder for innspill. Blant annet ble vi enige med ekstern veileder at candidates tabellen skulle flyttes ned og bli en del av application tabellen, og documents tabellen skulle kobles direkte til application.

Sprint 3

Hva var planen for sprinten?

Planen for sprint 3 var å starte på backend oppgaver. Det ble bestemt tidligere om å bruke C# ASP .NET Core⁵ og SQLite⁶ til databasen med EntityFramework⁷ fra Microsoft. Vi lagdelte programmet vårt etter Data Access Layer⁸ for å kunne få enklere tilgang til data som er lagret i databasen. Vi hadde seks repositories og de tilhørende controller som ble delt mellom alle medlemmene i gruppa.

⁵ [ASP .NET Core](#) er en rammeverk produsert av Microsoft for C#, F# og VisualBasic språkene. Rammeverket blir brukt for utvikling av webapplikasjoner. [12]

⁶ [SQLite](#) er en database engine som er basert på SQL. [13]

⁷ [EntityFramework](#) er en database rammeverk fra Microsoft, som gjør det enkelt å integrere database med applikasjonen. [14]

⁸ [DAL](#) er det laget i applikasjonens design arkitektur som har tilgang til databasen.

For denne sprinten var fokuset å jobbe med de funksjonelle krav til systemet som skulle bli gjort:

Søknad:

- Systemet skal gjøre det mulig å registrere kandidater til en stilling (søknader til stillingen).
- Systemet skal lagre endringer på status av en kandidat.
- Systemet skal lagre alle dokumenter tilhørende en kandidater.
- Systemet skal lagre et nytt dokument med endringer fra det originale dokumentet.
- Systemet skal hente data om en kandidat og alle kandidatene.

Jobb stilling:

- Systemet skal hente alle stillinger som er lagret.
- Systemet skal lagre en ny stilling.
- Systemet skal endre status til en stilling fra aktiv til arkivert
- Systemet skal hente data som er lagret av en stilling.

Lenke:

- Systemet skal lagre alle lenkene som tilhører en stilling
- Systemet skal hente alle lenkene som tilhører en stilling

Hva fullførte vi?

Blant de funksjonelle kravene som vi hadde, har gruppen klart å fullføre kravene Søknad og Jobb stilling. I tillegg har vi flyttet på pdf lagring ansvaret fra klient til server. Det gjorde vi for å optimalisere lagring og lesing av filene ved å redusere forflytning av stor tekst mellom lagene. En annen årsak var sikkerhet av filene, fordi om de var lagret på klient-siden da skulle de være lett tilgjengelig for alle som hadde nettsiden åpen. Det skulle vært en sikkerhetshull som kunne forårsake brudd på konfidensialitet ved å tilgjengeliggjøre data til uautoriserte klienter.

Hvilke utfordringer oppstod?

Utfordringen vi hadde under denne sprinten var lagring og henting av data til og fra databasen gjennom dataoverføringsobjekter.

Sprint 4

Hva var planen for sprinten?

Planen for sprint 4 var å starte med koblingen mellom frontend og backend og forbedre enhetstesting. Vi utførte enhetstester i programmet gjennom alle sprinter, men vi hadde mer fokus på det i denne sprinten på grunn av endringer i prosjektet. Derfor måtte enhetstestene samsvare med de endringene. Disse endringene handlet om modularitet, testbarhet og gjenbrukbarhet av tjenester og utvidelser som filbehandling, mappe behandling, JWT⁹, passord og anonym navnegenerator.

Hva fullførte vi?

I denne sprinten ble vi ikke ferdig med koblingen mellom frontend og backend ettersom enhetstesting tok lang tid og vi derfor utsatt dette til sprint 5. Gruppen implementerte testingskode for de ulike kontroller-metodene som sørger for at metodene virker som de skal.

Hvilke utfordringer oppstod?

I testingen hadde vi utfordringer med UserClaims¹⁰, fordi noe av API-endepunkter krever en innlogget bruker på systemet for å tilby riktig data eller funksjonalitet til

⁹ [JWT](#) står for Json Web Token. Det er en objekt som inneholder litt encodert data som brukes for å autentisere brukere.

¹⁰ [UserClaims](#) er encodert objekter som blir bevart i JWT objektet og senere blir de brukt for, autentisering og autorisering av brukeren.

sluttbruker. En annen problem i testingen var at testene kompromisset integritet av systemet ved å ha påvirkning på sensitive datafiler. Dette løste vi tilslutt med å mocke funksjonene som dannet dette problemet.

Sprint 5

Hva var planen for sprinten?

Planen for sprint 5 var å bli ferdig med api koblinger i første omgang. Etter det begynte vi med implementering av side restriksjoner og en forenklet epost tjener som informerer kandidaten, og gjør det mulig for administrasjonsbrukere å endre passord. I tillegg til det jobbet vi med klient valideringer.

Hva fullførte vi?

I denne sprinten klarte gruppen å fullføre alle oppgavene som var satt. Med det ble webapplikasjonen ferdig utviklet.

Hvilke utfordringer oppstod?

Noen av valideringene i former var litt kompliserte, men det ble løst etter at flere satt med det sammen og fant mulige løsninger som funket til slutt. Utenom dette så var det ikke andre utfordringer i denne sprinten.

Sprint 6

Hva var planen for sprinten?

Siden systemet var ferdig utviklet bestemte vi å begynne med integrasjonstesting av systemet. Målet med Integrasjonstesting er å teste både API laget og datatilgangslaget.

Hva fullførte vi?

I denne sprinten klarte vi å automatisere testene for integrasjonstesting ved hjelp av xUnit.net¹¹ på C#. Deretter testet vi programmet manuelt.

Hvilke utfordringer oppstod?

Utfordringene i denne sprinten var initialiseringen av data, database og filsystemet som tok oss litt mer tid enn det var planlagt. Men vi klarte å nå målene for denne sprinten.

2.5 Endringer og utfordringer

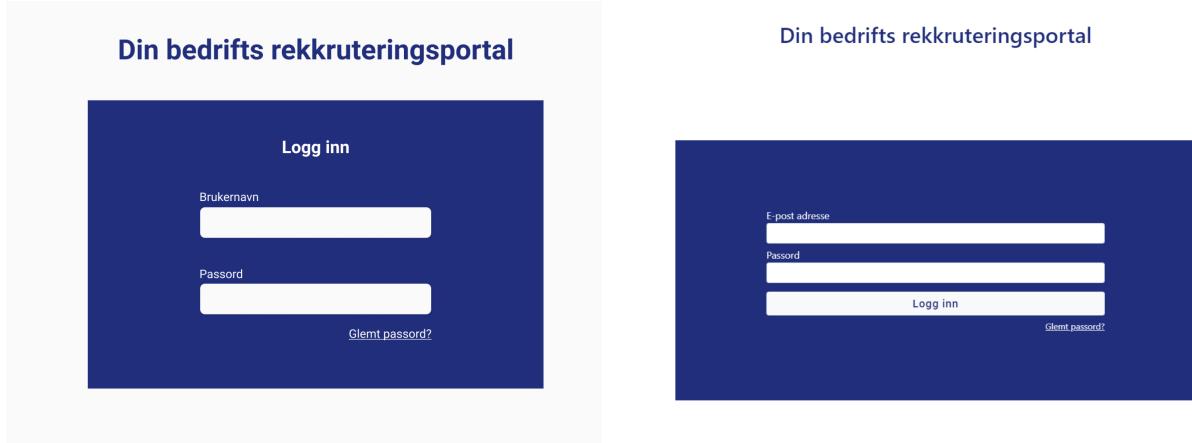
2.5.1 Mappe Restrukturering

Under utviklingen av applikasjonen var det nødvendig med en del omstrukturering av mapper, mens gruppen videreutvikler applikasjonen, ble flere filer og mapper opprettet. Dermed for å forhindre tap av produktivitetstid og dårlig finnbarhet av filer, restrukturerer vi filsystemet når det er nødvendig. Dette reduserer risikoen at vi bruker feil dokument og gjør det digitale arbeidsområdet vårt enklere å administrere.

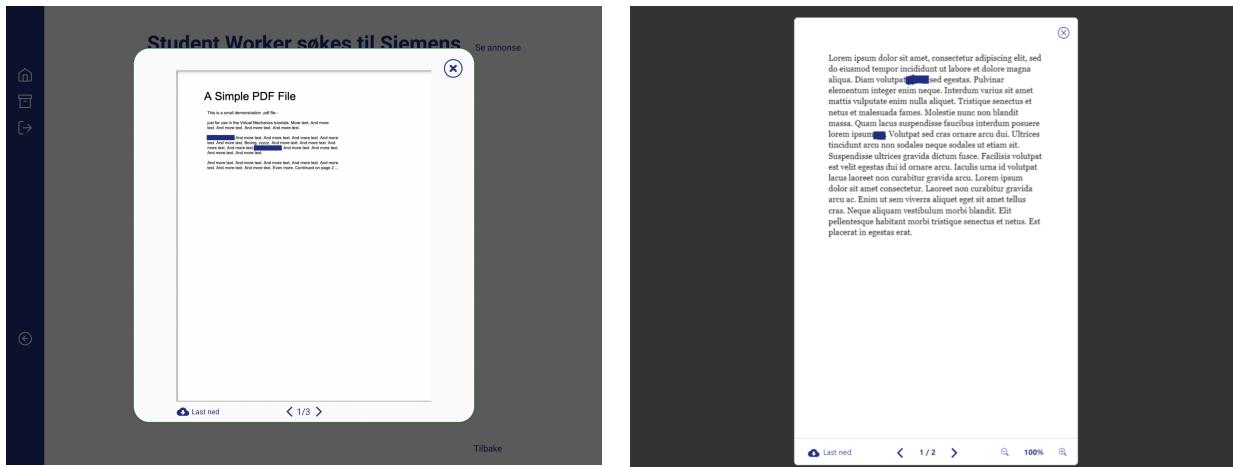
2.5.2 Designendringer

Denne seksjon inneholder designendringer som ble gjort under applikasjonsutviklingen. Figurene under viser før og etter endringer av designet.

¹¹ [xUnit.net](#) er en bibliotek i C# språket som brukes for automatisering av tester. [15]



Figur 2.5.2a: Innloggingsside endringer



Figur 2.5.2b: PDF-Viewer endringer

2.5.3 Rammeverk Oppgradering

Frontend-rammeverket gruppen har bestemt å bygge applikasjonen på er Angular¹².

Når applikasjonen ble først initialisert, startet gruppen med en boilerplate av Angular på versjon 8. Sammenlignet med den nyeste versjonen av Angular som er 13, kom

¹² Angular er en TypeScript rammeverk for utvikling av webapplikasjoner utviklet av Google.

versjon 8 med et par problemer. Noen av NPM¹³ pakkene som vi trenger var ikke kompatibel med denne versjonen. Gruppen klarte å løse dette ved å oppgradere rammeverket til versjon 12 som for øyeblikket er den mest stabile versjonen. I forhold til funksjonalitet og sikkerhet, er det ikke en stor forskjell mellom versjon 12 og 13.

2.6 Konklusjon av prosessen

2.6.1 Avvik i forhold til fremdriftsplan

Det har ikke vært noen store avviker i forhold til fremdriftsplanen. De avvikene vi har hatt har vært at noen av oppgavene i sprintene ble flyttet til neste sprinter, og endringer i prioritering. En annen oppgave som vi ikke har rukket å løse var kobling av email tjenesten med SMTP (Simple Mail Transfer Protocol).

2.6.2 Konklusjon og refleksjoner

Gjennom utviklingen av systemet, har gruppen lært veldig mye om utviklingen av en applikasjon ved bruk av .Net og Angular med REST API-er teknologier. Vi har møtt utfordringer som gruppen har både løst sammen og har fått hjelp til fra oppdragsgiver. Vi har også lært mye om smidig arbeidsmetodikk, hvordan man kommuniserer og diskuterer med oppdragsgiver, og hvordan man jobber effektiv med å utvikle en applikasjon, som blant annet prioritering av oppgaver.

¹³ [NPM](#) (Node Package Manager) står for Node pakke behandler, det hjelper med nedlasting og organisering av nødvendige pakker. [16]

Kapittel 3

Produktdokumentasjon

Innhold

- 3.1 Introduksjon til ARecruit - Anonymisert rekrutteringssystem
- 3.2 Produktspesifikasjoner
- 3.3 Frontend
- 3.4 Backend
- 3.5 Sikkerhet
- 3.6 Filstruktur
- 3.7 Konklusjon

3.1 Introduksjon til ARecruit - Anonymisert rekrutteringssystem

ARecruit er en anonymisert rekrutteringssystem for bedrifter, og skal gjøre det mulig for kandidater å være anonyme når de søker til stillinger frem til de blir invitert til intervju. Det funker ved at en kandidat fyller inn kontaktopplysninger, og laster opp CV og søknad.

Når søknaden er sendt inn, vil den ikke bli sendt til beslutningstaker (HR) umiddelbart, men til en som har rollen som inspektør. En inspektør funker som en slags mellomledd eller knytningsperson mellom kandidaten og beslutningstaker. Den har alle opplysninger om kandidaten, og deres jobb er å anonymisere søknaden til kandidaten og sende den til beslutningstakeren.

Når inspektør har ferdig anonymisert søknaden til kandidaten vil den bli sendt videre til beslutningstaker, og der så kan beslutninstakeren kun se de relevante opplysningene for stillingen som erfaring, utdanning osv. Når beslutningstaker har tatt en avgjørelse av kandidaten, blir den sendt videre igjen til inspektør, og inspektør varsler kandidaten om avgjørelsen.

3.2 Produktspesifikasjoner

3.2.1 Kravspesifikasjon

En kravspesifikasjon er en oversikt over ønskede funksjonalitet, egenskaper og andre krav som applikasjonen skal ha. Disse kravene blir ofte delt i ikke-funksjonelle og funksjonelle krav. Ikke funksjonelle krav handler mer om kvaliteten til et system eller applikasjon, mens funksjonelle krav omhandler funksjoner og egenskaper til et applikasjon. Hva som er etterspurt av kunden/brukeren at applikasjonen skal utføre,

kommer også inn i funksjonelle krav. Gruppens funksjonelle krav har vi fått fra oppdragsgiver, mens ikke-funksjonelle er krav som gruppen sammen med ekstern veileder har blitt enige om. [17]

3.2.1.1 Ikke-funksjonelle krav

- Det skal være lett og enkelt å utføre oppgavene
- Det skal ikke ta lang tid å utføre oppgavene
- Applikasjonen skal ha et enkelt design og være responsivt
- Applikasjonen skal være brukervennlig for kandidat for å søke til stillinger, og inspektør for å anonymisere søknader.
- Personvernet til kandidater skal opprettholdes
- Tilegne en anonym id automatisk til kandidaten når en søknad blir sendt.
- Systemet skal kunne ha universell utforming
- Systemet skal opprettholde konfidensialiteten, integriteten og autentisiteten.

3.2.1.2 Funksjonelle krav

Beslutningstaker

- Se anonymiserte søknader som er sendt fra inspektør.
- Markere status til søker: Enten Godkjent eller Ikke Godkjent. Søknaden Skal få automatisk en “venter på godkjenning” status når det ikke er tildelt status fra beslutningstaker.
- Lage nye stillingsannonser som inspektør får tilgang til og en kandidat kan søke til.

- Vil se en liste eller oversikt over søknadene som har blitt godkjent fra beslutningstaker.
- Vil se en liste eller oversikt over søknadene som venter på godkjenning fra beslutningstaker.
- Vil se en liste eller oversikt over søknadene som ble ikke godkjent fra beslutningstaker.
- Vil ha en oversikt over arkivert stillinger og søknader

Inspektør

- Se søkeres søknader som er sendt inn fra kandidat.
- Bruke anonymiseringsverktøy: For å anonymisere søkeres CV og søknadsbrev som er sendt inn før den blir sendt videre til beslutningstaker.
- Ha tilgang til original kopier samt anonymiserte kopier under kandidatens ID. Dette for å sørge for at inspektør fortsatt har kontaktinformasjonen til en kandidat.
- Sende videre en ferdig anonymisert søknad til beslutningstaker.
- Vil se en liste eller oversikt over søkeres søknadene som har blitt godkjent fra beslutningstaker.
- Vil se en liste eller oversikt over søkeres søknadene som venter på godkjenning fra beslutningstaker.
- Vil se en liste eller oversikt over søkeres søknadene som ble ikke godkjent fra beslutningstaker.
- Vil ha en oversikt over arkivert stillinger og søkeres søknader

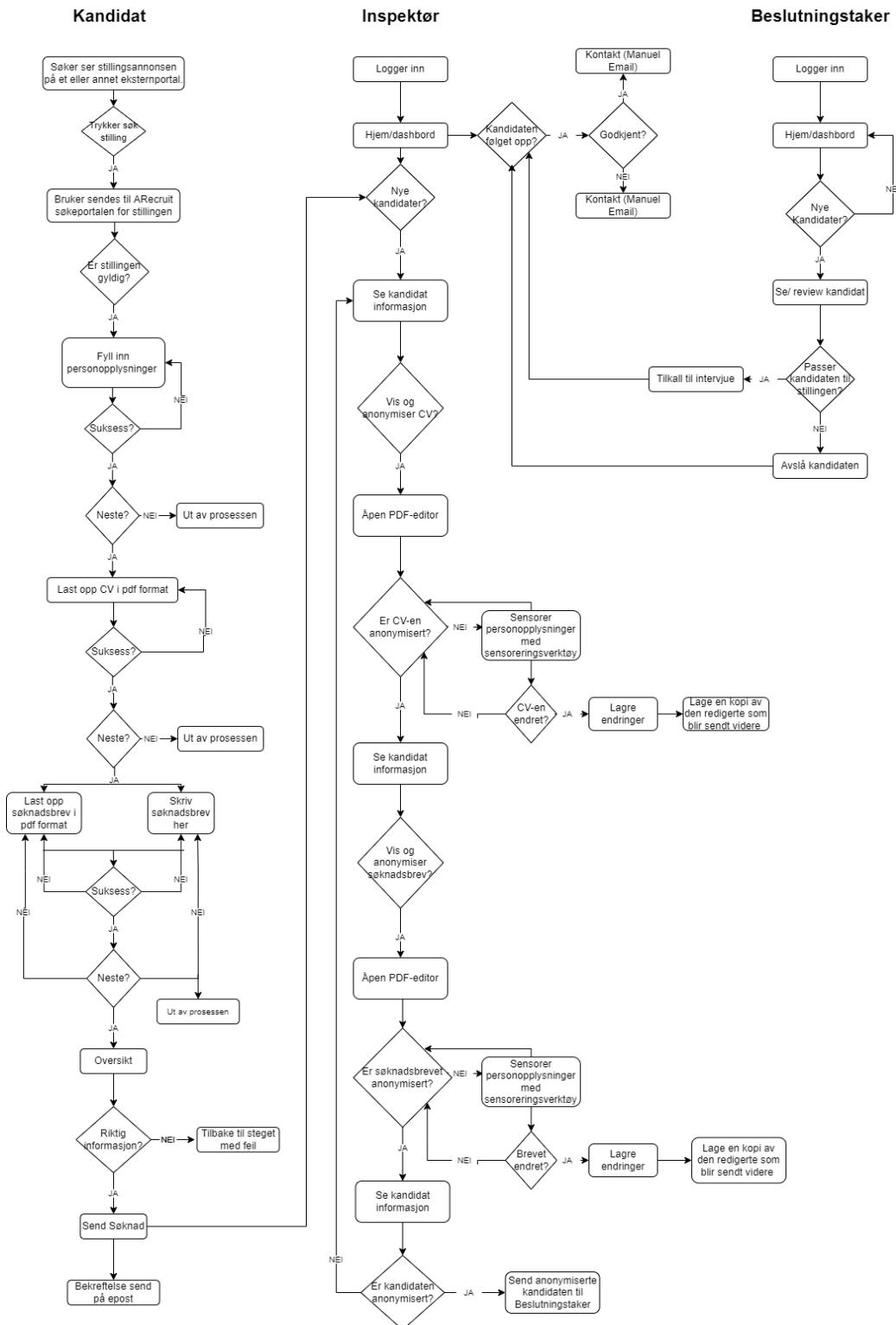
Kandidat

- Søke på en stilling: Føre inn personalia og legge ved CV og Søknadsbrev (eller skrive søknadsbrevet selv), for så deretter sende søknaden

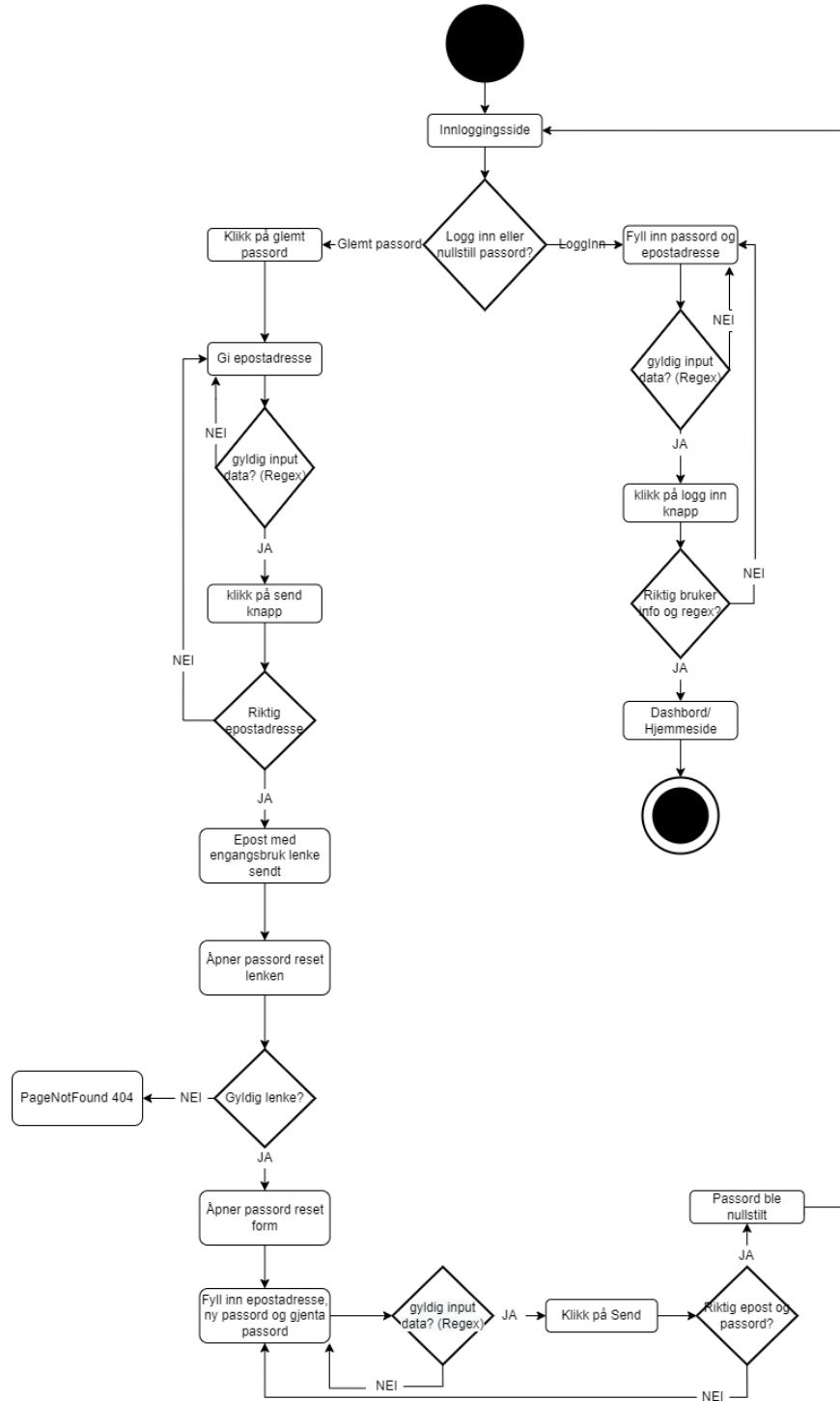
3.2.2 Aktivitetsdiagram

Aktivitetsdiagram er en UML¹⁴ diagram som viser grafisk fremstilling av aktiviteter og handlinger etter valgene bruker gjør i systemet. *Figur 3.2.2-A* er en aktivitetsdiagram av systemet, mens *Figur 3.2.2-B* er om autoriseringsprosessen. Begge diagrammene viser alle mulige utfall av aktiviteter, på *Figur 3.2.2-A* under aktøren *Beslutningstaker*, er et av aktivitetene å se søkeren til en kandidat og ta en avgjørelse. Denne aktiviteten har to utfaller, enten ”*Tilkall til intervju*” eller ”*Avslå kandidaten*”.

¹⁴ [UML](#) står for Unified Modeling Language som blir ofte bruk for presentering av en software før den skal utvikles.



Figur 3.2.2-A: Aktivitetsdiagram av systemet



Figur 3.2.2-B: Flytdiagrammen av Autoriseringsprosessen

3.3 Frontend

Denne seksjonen består av teknologier, illustrasjoner, komponenter og logikker som er implementert og brukt på frontend siden av applikasjonen.

3.3.1 Teknologier og verktøy

3.3.1.1 Angular

Angular er en gratis og en åpen kildekode rammeverk som baserer seg på TypeScript¹⁵. Det blir brukt for å utvikle en ensideapplikasjon (SPA), som vil si at nettsiden har kun én side som blir oppdatert med nytt innhold mens den kommuniserer med brukeren, istedenfor å lage flere sider. [18]

3.3.1.2 Bootstrap

Bootstrap¹⁶ er en åpen kildekode rammeverk som skal gjøre implementering av responsive webapplikasjoner mer enklere og effektivt. Dette er spesielt viktig da webappen skal være den samme for både mobil, nettbrett og pc, men hvordan et innhold blir vist er det som skal tilpasses seg etter enhetens størrelse. Bootstrap har forskjellige maler man kan bruke, som knapper, skjemaer, navigasjonsmenyer og mer. [19]

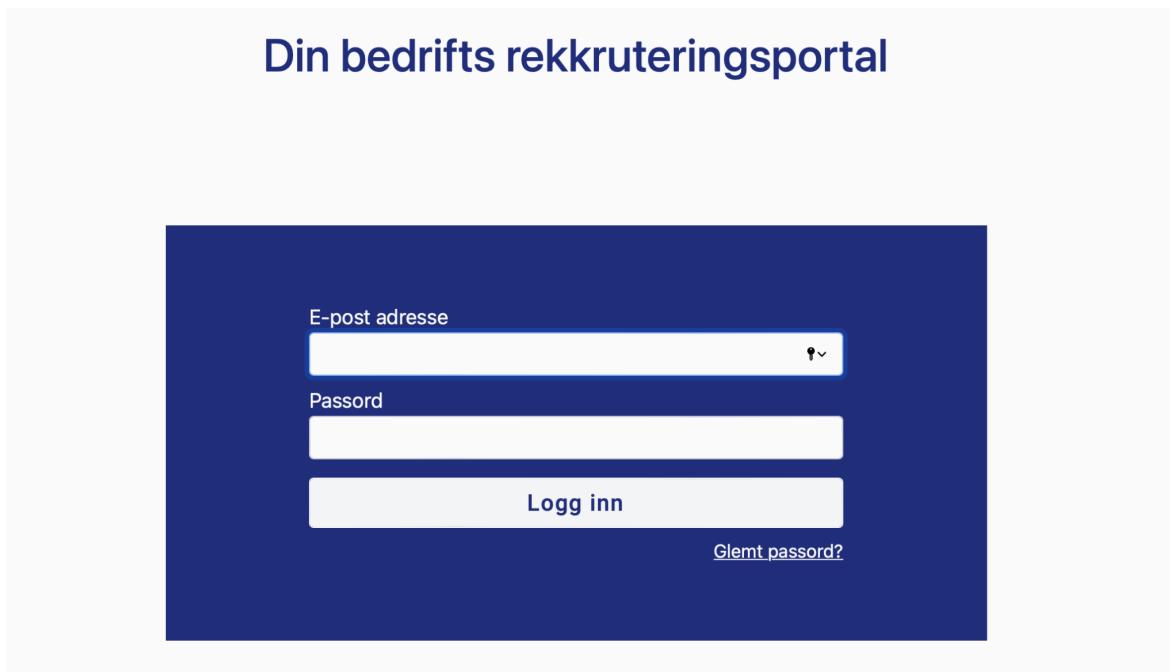
¹⁵ [TypeScript](#) er et objektorientert programmeringsspråk.

¹⁶ [Bootstrap](#) er en enkelt og raskt rammeverk for design og oppbygning av nettsider.

3.3.2 Hovedsider

3.3.2.1 Innloggingsside

Innloggingssiden er for admin brukere i plattformen. En admin må gå gjennom en autorisasjon før brukeren får tilgang til deres side. Dette gjøres ved denne siden hvor admin skriver mailadresse og passord.



Figur 3.3.2.1 Logg inn side

3.3.2.2 Passord reset siden

Plattformen har en tilbakestill passord funksjon på logg inn siden. Hvis admin har glemt sitt passordet så kan admin trykke på "Glemt passord?" lenken og oppgi epostadresse den er registrert med. Da vil en epost med en link bli sendt til brukeren og brukeren kan bruke denne linken til å tilbakestille passordet sitt.

Din bedrifts rekruteringsportal



Figur 3.3.2.2-A Tilbakestillingslenke blir sendt på epost

Din bedrifts rekruteringsportal



Figur 3.3.2.2-B Tilbakestilling av passord side

3.3.2.3 Stilling dashboard for admin

Stilling dashboard er en komponent som viser oversikt over alle aktive og ikke aktive stillinger. Hvis admin er en beslutbingstaker så har adminen tilgang til å legge til flere nye stillinger. Som det er vist på Figur 3.3.2.3.



Figur 3.3.2.3 Stillingsdashboard for beslutningstaker

3.3.2.4 kandidat dashboard for admin

Kandidat dashbordet skal vise en oversikt over anonymiserte og ikke anonymiserte kandidater som har søkt på stillingen etter hvem admin er.

The screenshot shows a candidate dashboard titled "Deltidsansatt fullstack utvikler". At the top right is a link "Se annonsen". Below it is a section titled "Alle kandidater" with four filter buttons: "Alle", "Godkjente", "Avviste", and "Avventer". There are four cards representing candidates:

- Card 1: "stolte_ildkjuke" (Godkjent) - green border
- Card 2: "hult_djevelklo" (Ikke godkjent) - red border
- Card 3: "fin_kveke" (Venter på godkjenning) - orange border
- Card 4: "gyllen_gjeldkarve" (Godkjent) - green border

At the bottom right is a link "Arkiver annonse".

Figur 3.3.2.4-A: Beslutningstaker sin kandidat dashboard

The screenshot shows a candidate dashboard titled "Deltidsansatt fullstack utvikler". At the top right is a link "Se annonsen". Below it is a section titled "Nye, Ikke vurderte kandidater" containing one card:

- Card 1: "Martin Henriksen nobelt_hestehov" (Ikke anonymisert) - red border

Below this is a section titled "Alle kandidater" with four filter buttons: "Alle", "Godkjente", "Avviste", and "Avventer". There are four cards representing candidates:

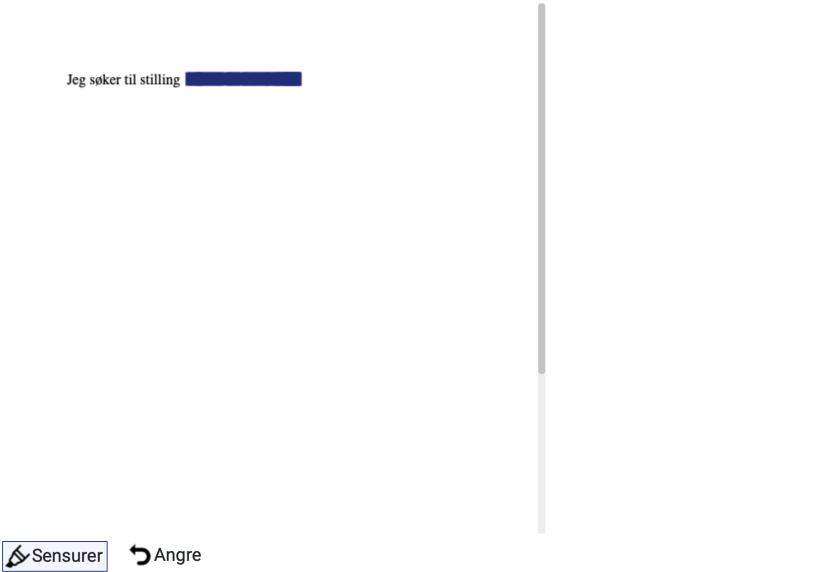
- Card 1: "stolte_ildkjuke" (Godkjent) - green border
- Card 2: "hult_djevelklo" (Ikke godkjent) - red border
- Card 3: "fin_kveke" (Venter på godkjenning) - orange border
- Card 4: "gyllen_gjeldkarve" (Godkjent) - green border

At the bottom right is a link "Arkiver annonse".

Figur 3.3.2.4-B: Inspektør sin kandidat dashboard

3.3.2.5 PDF-editor for admin

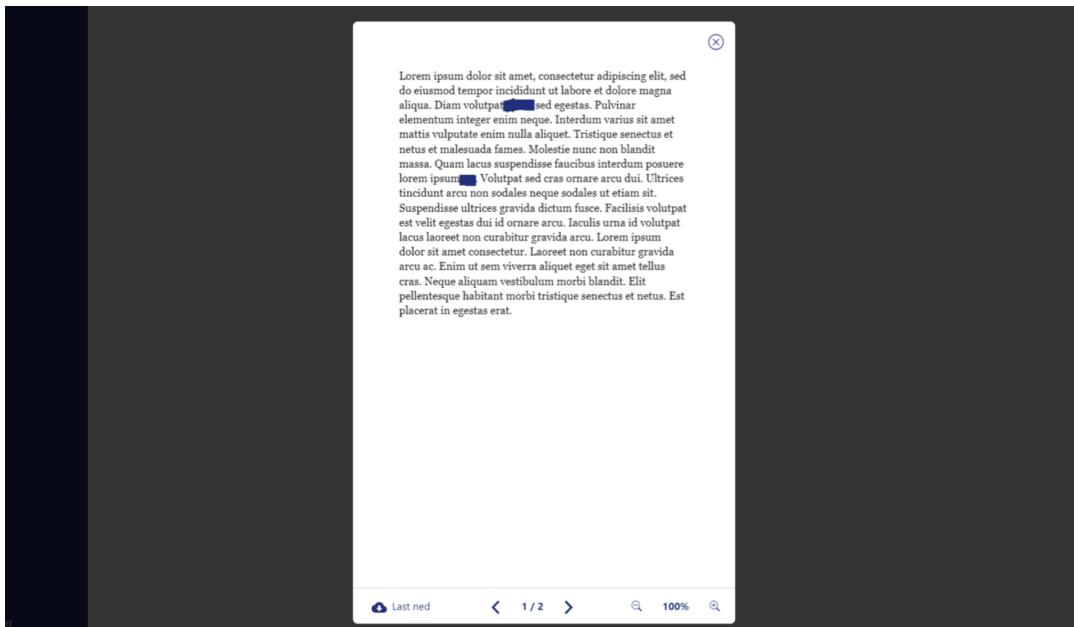
Denne PDF editoren kan kun bli aksessert av inspektøren. Det er der inspektøren anonymiserer søknadene og cv'ene som kommer inn før de blir videresendt til beslutningstaker.



Figur 3.3.2.5: Pdf-editor som kun inspektør har tilgang på

3.3.2.6 PDF-viser for admin

Adminene med rollen som beslutningstaker kan aksessere denne siden. Med pdf-vieweren kan de lese, laste ned og bla gjennom den anonymiserte cv-en og søknaden til en kandidat.



Figur 3.3.2.6: Pdf-Viewer som kun beslutningstaker har tilgang på

3.3.2.7 Søknadsportal

Dette er den eneste siden kandidaten har tilgang til, og er satt sammen av flere komponenter. Kandidat leser stillingsannonsen på eksterne sider, og på disse eksterne sidene er det lenket til denne søkerportalen som tilhører stillingen hvor kandidaten kan søke på stillingen. Denne siden er satt sammen av tre skjemaer man må fylle før søkeren blir sendt, disse er personaliaskjema, cv opplastning og skjema for søkerbrev.

The screenshot shows a recruitment application form titled "Sommerjobb utvikler". At the top right is a link "Se annonsen". Below the title are four circular icons with labels: "Personalia" (person icon), "CV" (cv folder icon), "Søknadsbrev" (document icon), and "Fullfør" (checkmark icon). The "Fullfør" icon is highlighted with a blue border. The form fields include:

- "Hva heter du?" input field containing "Ola Norman"
- "E-post adresse" input field containing "olanorman@ways.com"
- "Telefonnummer" input field containing "12345678"
- "Fødselsdato" input field containing "20.04.2000"

A large empty rectangular area is positioned below these fields. At the bottom right is a blue-outlined button labeled "Neste".

Figur 3.3.2.6-A: Personalanskjema (første steg i søkerportalen)

3.3.3 Komponenter

Applikasjonens brukergrensesnitt er bygget opp av små komponenter. Disse komponentene kan endres og oppgraderes uten at de påvirker andre komponenter som ikke er relatert til dem. Separasjon øker gjenbrukbarheten og testbarheten. En komponent er kun ansvarlige for sine egne indre funksjonaliteter og bør ikke være opptatt av den generelle tilstanden til applikasjonen.

1

2

3

4

Student Worker søker til Siemens [Se annons](#)

Nye, ikke vurderte kandidater

Rolig Seilkar
Ikke anonymisert

Rolig Seilkar
Ikke anonymisert

Rolig Seilkar
Ikke anonymisert

Alle kandidater

Alle Godkjente Avviste Avventer

Rolig Seilkar
Godkjent for videre oppfølging

Rolig Seilkar
Godkjent for videre oppfølging

Rolig Seilkar
Godkjent for videre oppfølging

Rolig Seilkar
Venter på godkjenning

Rolig Seilkar
Venter på godkjenning

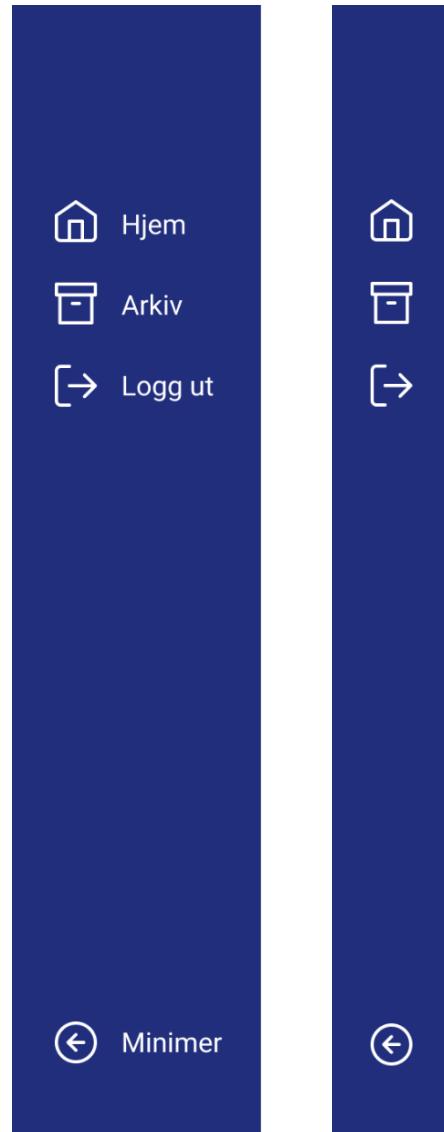
Rolig Seilkar
Ikke godkjent

Arkiver annons

Figur 3.3.3: Et eksempel på en side som er bygget opp av 4 komponenter.

3.3.3.1 Navigasjonskomponent

Navigasjonskomponenten er en del av admin sin hovedsiden. Adminen kan navigere seg *Hjem* hvor en liste av aktive stillinger vises, *Arkiv* hvor alle arkiverte stillinger befinner seg og *Logg ut*. Nederst finnes det et ikon som adminen kan trykke på for å maksimere eller minimere sidefeltet.



Figur 3.3.3.1-A:
Maksimert nav

Figur 3.3.3.1-B:
Minimert nav

3.3.3.2 Kandidat sammendrag komponenter

Personalia komponenten er en komponent som er brukt av to aktører i applikasjonen: *kandidaten* og *inspektøren*. Denne komponenten virker som en sammendrag for begge aktørene. Det viser all informasjon om en kandidat. I tillegg til dette, har inspektøren tilgang til en liten prompt under, hvor de kan oppgi om en kandidat er ferdig anonymisert eller ikke. Denne prompten viser ikke hos kandidaten.

Personalia

Anonym ID: Smilende Dans
Navn: Bjørn-Ivar Skuggen
Telefonnummer: +47 476 61 614
E-post: bjorn.ivar.skuggen@ways.no

CV
[CV_Smilende_Dans.pdf](#)

Søknadsbrev
[Brev_Smilende_Dans.pdf](#)

Er kandidaten ferdig anonymisert?

[JA](#)

Figur 3.3.3.2-A:
Personalia komponent

Når adminen som er logget inn er *beslutningstaker*, er det denne komponenten som er vist. Det inneholder anonymisert informasjon om en kandidat. Med denne komponenten, har beslutningstakeren evnen til å godkjenne eller avvise en søknad.

mandig_ormetelg

cv
[CV_mandig_ormetelg_anonym.pdf](#)

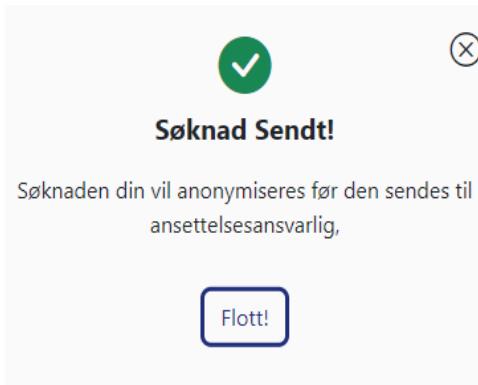
Søknadsbrev
[Søknad_mandig_ormetelg_anonym.pdf](#)

Tilkal til intervju

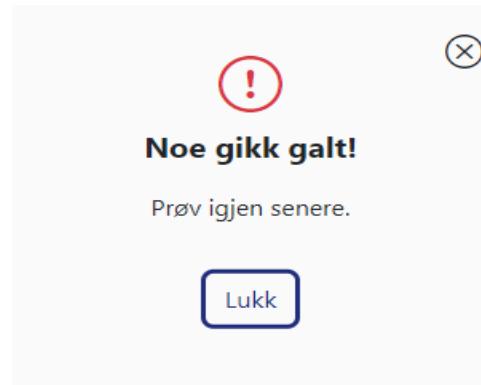
Avslå søknad

Figur 3.3.3.2-B:
Profil komponent

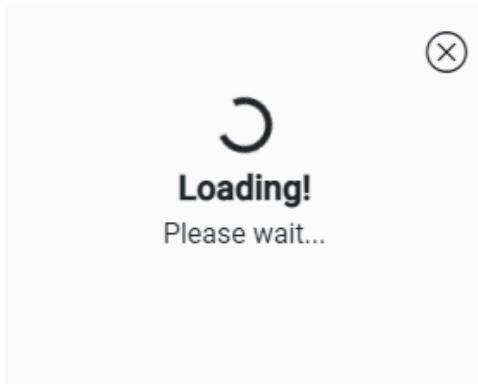
3.3.3.3 Alert komponent



Figur 3.3.3.3-A:
Suksessmelding



Figur 3.3.3.3-B:
Advarsel



Figur 3.3.3.3-C:
Innlasting



Figur 3.3.3.3-D:
Konfirmasjon

Alert komponenten sin rolle er å varsle brukere om informasjon som krever brukerens umiddelbar oppmerksomhet. På applikasjonen er denne komponenten brukt til å vise suksessmeldinger, advarsler og prompter. Denne komponenten har 6 typer og er definert som *enum* i applikasjonen: Suksess, advarsel, innlasting, informasjon, tilpasning og konfirmasjon.

3.3.3.4 Filopplastningskomponent



Denne komponenten er ansvarlig for opplasting av filer. Det er kun tilstede i kandidatens søknadsportalen. Det finnes på to skjemaer: en for opplastning av en cv og en for opplastning av et søkeradsbrev.

Figur 3.3.3.4:
File Upload

3.3.3.5 Kort komponenter

Kandidat kortet er lastet inn med kandidatens navn og søkerad status, mens ad kortet inneholder tittelen til en stilling. Disse komponentene er gjengitt på riktig foreldre komponenter basert på noen attributter.

Kandidat kortet har tre farger: rød som representerer at kandidaten er ikke anonymisert eller ikke godkjent, gul som betyr at kandidaten venter på godkjenning og grønn som representerer at kandidaten er godkjent.



Figur 3.3.3.5-A: Kandidat kortene

Student Worker søkes til Siemems

Figur 3.3.3.5: Stillingsannonse kort

3.3.4 Modeller

Modeller på frontend er rammer for data som blir sendt til server eller levert til klient.

Disse modellene sørger for riktig oppbygning av objekter.

3.3.5 Route guards og resolvers

```
const routes: Routes = [
  {path: '', redirectTo: 'auth/login', pathMatch: 'full' },
  {path:'404', component: PageNotFoundComponent},
  {path:'auth', redirectTo: "404"},
  {path:'admin', redirectTo: "404"},

  {
    path: 'admin',
    runGuardsAndResolvers: 'always',
    canActivate: [ AuthGuard ],
    component: AdminComponent,
    children: [
      {path:'ad/create', component: AddNewPositionComponent, canActivate: [ ManagerRoleGuard ]},
      {path:'ads/:state', component: AdsDashboardComponent, resolve: { ads: AdResolver }},
      {path:'ad/:id', redirectTo:"404"},
      {path:'ad/:id/candidates', component:CandidateDashboardComponent, resolve: { applications: CandidateResolver }},
      {path:'anonymous/candidate/:id/ad/:adId', component: CandidateSummaryComponent, canActivate: [ ManagerRoleGuard ]},
      {path:'candidate/:id/ad/:adId', component: PersonaliaComponent, canActivate: [ InspectorRoleGuard ]},
      {path:'pdf-editor/pdf/:docId/application/:appId/:adId', component: PdfEditorComponent, canActivate: [ InspectorRoleGuard ]},
      {path:'pdf-editor/pdf/:docId/application/:appId/:adId/anonymous', component: PdfEditorComponent, canActivate: [ InspectorRoleGuard ]},
      {path:'pdf-viewer/:id/application/:appId/:adId', component: PdfViewerComponent, canActivate: [ ManagerRoleGuard ]},
    ]
  },
  {path: 'auth', component: AuthComponent , children:[
    {path: 'login', component: LoginComponent},
    {path: 'confirm-email', component: ConfirmEmailComponent},
    {path: 'resetPassword/:id', component: PasswordResetComponent}
  ]}
];
{path:'apply-for/ad/:id', component: CandidateComponent},
{path:'**', redirectTo: "404"},

];
```

Figur 3.3.5: App routes

Angular *route guards* gir eller fjerner tilgang til bestemte sider av applikasjonen avhengig av om en bruker har de riktige tilgangsrettighetene. *Figur 3.3.5* viser alle rutene applikasjonen har. En av rutene er **hostname/admin/ad/create**. Denne ruten er beskyttet av *AuthGuard* og *ManagerRoleGuard*. For å få tilgang til *AdNewPositionComponent*, *AuthGuard* krever at brukeren må være logget inn og *ManagerRoleGuard* krever at når brukeren har logget seg inn, må de ha Manager rollen omtalt som beslutningstaker. Hvis disse kravene er oppfylt, kan brukeren aksessere siden.

Om kravene av *AuthGuard* ikke er oppfylt, blir brukeren omdirigert til logginn siden. Men hvis *AuthGuard* sine krav er oppfylt men ikke *ManagerRoleGuard*, blir brukeren omdirigert til en komponent som krever ikke en rolle basert guard. Brukeren er dermed omdirigert til *AdsDashboardComponent*, en komponent hvor alle logget inn brukere har tilgang på uansett rollen de har.

Route resolver er måten å forhåndshente data før en komponent er lastet inn. Ruten **hostname/admin/ads/:state** er konfigurert med *AdResolver*. Dette vil hente alle stillingsannonserne først før *AdsDashboardComponent* er lastet inn. Med *route resolvers*, er komponenter sørget for at de inneholder alle data de trenger før de er presentert til brukeren.

3.3.6 Services

Hovedmålet med Services er å organisere og dele logikk, data og metoder mellom komponentene i en Angular applikasjon. De er implementert vanligvis gjennom avhengighetsinjeksjon.

3.3.6.1 HTTP Services

HTTP¹⁷ tjenesten er tjenesten som kobler klienten med serveren. Rest api-er er implementert på denne tjenesten. Ved hjelp av api-endepunktene som er definert i bakenden, kan klienten utføre CRUD operasjonene¹⁸ som å hente, sende og oppdatere dataene i databasen.

```

@Injectable({providedIn:'root'})
export class AdService extends DataService{
  constructor(http:HttpClient) {
    super("ad",http);
  }
}

@Injectable({providedIn:'root'})
export class ApplicationService extends DataService{
  constructor(http:HttpClient) {
    super('application',http)
  }
}

@Injectable({providedIn:'root'})
export class CandidateService extends DataService{
  constructor(http:HttpClient) {
    super('application/candidate',http)
  }
}

@Injectable({providedIn:'root'})
export class DocumentService extends DataService{
  constructor(http:HttpClient) {
    super('document',http)
  }
}

@Injectable({providedIn:'root'})
export class LinkService extends DataService{
  constructor(http:HttpClient) {
    super('link',http)
  }
}

```

Figur 3.3.6.1: HTTP Services

3.3.6.2 Authentication Services

Auth tjenesten håndterer logginn prosessen. Den får en gyldig token fra serveren ved vellykket autentisering. Auth tjenesten benytter *JwtTokenService* som utvider det til å dekode og sjekke utløpsdatoen til en token. Tokenen er lagret i nettleserens lokal lagring etter en vellykket autentisering og deretter fjernet når brukeren logger seg ut.

¹⁷ [HTTP](#) står for Hypertext Transfer Protocol er en protokoll i applikasjonslaget i internett protokollen.

¹⁸ [CRUD](#) operasjonene er forkortelse for operasjonene opprett, les, endre og slett.

Token payload-en inneholder tre opplysninger: rollen, e-postadressen og brukernavnet til en admin. Rollen er brukt som en betingelse for å sjekke om en admin kan aksessere en bestemt side på applikasjonen eller ikke. Dessuten er e-posten og brukernavnet brukt i *InformServices* til å sende e-post til kandidater når en admin har ferdig vurdert en søknad.

3.3.6.3 Inform Candidate Services

Hovedhensikten med Inform tjenesten er å sende en e-post til en kandidat etter en vellykket innlevering, avvisning og godkjenning av deres søknad. Denne tjenesten er konfigurert med api-endepunktene som oppdaterer en data attributt i databasen for å holde styr på om en kandidat er kontaktet eller ikke.

3.3.6.4 Alert Services

Alert tjenesten har ansvaret til å åpne, lukke, tilpasse og gjengi alert komponenten og dens innhold. Det er injisert i andre komponenter for å vise varslinger eller meldinger som krever brukerens oppmerksomhet.

```
@Injectable({
  providedIn: 'root'
})
export class AlertService {
  private subject = new BehaviorSubject<any>({});

  constructor() { }

  openAlert(alertData: Alert) {
    this.subject.next(alertData);
  }

  closeAlert() {
    this.subject.next(null);
  }

  getAlert(){
    return this.subject.asObservable();
  }
}
```

Figur 3.3.6.4: Alert tjenesten

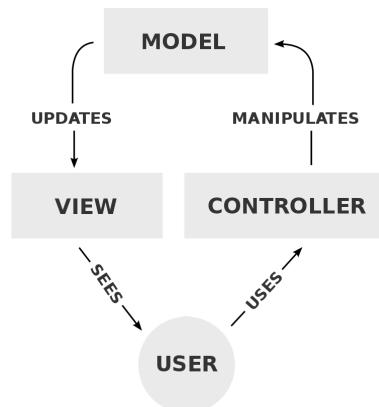
3.3.6.5 Password Services

Applikasjonen gir funksjonaliteten til å oppdatere passordet til adminene gjennom denne tjenesten. Et krav om en gyldig e-post må være oppfylt. Ved en gyldig e-post, får en admin en lenke der de kan oppgi et nytt passord. Etter en vellykket passord oppdatering, blir lenken ubruklig.

3.4 Backend

I denne seksjonen presenterer vi de delene som er implementert og brukt for å bygge opp bakenden.

For applikasjonens backend design brukes det et MVC mønster. MVC mønsteret eller modellen er en forkortelse for modell, view og kontroller. I dette mønsteret ser sluttbruker viewet som er frontend applikasjonen. Kontroller kobler view og backend logikken sammen. Deretter manipulerer og behandler modellen data. Eksemplet på den er dataaksess laget og filesystem tjenester som lagrer og modifiserer data etter sluttbrukers krav. I tillegg til det, utfører modellen interne oppgaver som er relatert til eposttjenester og sikkerhet. [20]



Figur 3.4: MVC software design pattern diagram [21]

3.4.1 Teknologier og verktøy

Følgende seksjoner presenterer de forskjellige teknologiene vi har brukt på bakenden.

3.4.1.1 C# ASP.NET Core

ASP.NET Core er en platform og en åpen kildekode rammeverk for utvikling av moderne applikasjoner for forskjellige plattformer som web og mobil. Det er utviklet av Microsoft og er en av de raskeste og mest brukte rammeverkene som er tilgjengelig. [\[12\]](#)

3.4.1.2 Entity Framework

Entity framework er en åpen kildekode rammeverk som støttes av microsoft, og gjør at vi slipper å skrive sql spørresetninger, og heller aksesserer databasen via collections isteden. Det resulterer til en bedre sikkerhet når man jobber med databaser, for det er vanskeligere å utføre angreper mot databasen. I tillegg tilbyr rammeverket en enklere migrasjon av databasen om applikasjonen skal videreføres i en senere anledning. [\[14\]](#)

3.4.2 Web Layer

Denne seksjonen inneholder detaljer om kontroller delen i MVC¹⁹ mønstret. Det består av kontrollerer, dataoverføring objekter og API-kontrakter.

3.4.2.1 Controllers

Kontrollere er koblingspunktet mellom klienten(sluttbrukere på web) og serveren. Kontroller klassene hører på HTTP forespørsel fra klienten som inneholder en

¹⁹ [MVC](#) står for Model, View, Controller

overskrift og payload. Overskriften består av en API Route og en Bearer Token²⁰, men payload-en er data i form av JSON²¹.

Etter at en HTTP forespørsel sendes, utfører metoden det som klient har bedt om ved hjelp av andre tjenester som ligger på serveren. Til slutt returnerer metoden et HTTP respons.

Liste over Controllers:

Controller Navn	Beskrivelse
AdController.cs	Stillingskontroller er ansvarlig for CRUD operasjon for stillingsentiteter. Klassen er komponert av IAdRepository.cs og filsystem tjenester.
AdminController.cs	Adminstrasjonskontroller er ansvarlig for CRUD operasjon for Admininstrasjonsentiteter. Klassenen er komponert av IAdminRepository.cs og eposttjenester.
ApplicationController.cs	Søknadskontroller er ansvarlig for CRUD operasjon for søkeresentiteter. Klassenen er komponert av IApplicationRepository.cs og filsystem tjenester.
DocumentController.cs	Dokumentkontroller er ansvarlig for CRUD operasjon for dokumentsentiteter. Klassenen er komponert av IDocumentRepository.cs og filsystem tjenester.

²⁰ Bearer Token er en HTTP ordning som brukes mest for sikkerhet av rutene.

²¹ JSON er Javascript objektnotasjon som er en fil og data format. Det blir ofte brukt i veksling av data mellom flere systemer. [22]

LinkController.cs	Lenke kontroller er ansvarlig for henting av lenker fra databasen. Klassen er komponert av ILinkRepository.cs.
InformCandidateController.cs	Informer kandidat kontroller er ansvarlig for å sende epost til kandidater. Klassen er komponert av eposttjenester og IApplicationRepository.cs.
AuthController.cs	Autentiseringskontroller er ansvarlig for identitetsbekrftelse av bruker. klassen er komponert av passord utvidelser, JWT tjenester og IAdminRepository.cs.

3.4.2.2 Dtos (Data Transfer Objects)

Dataoverføringobjekter er objekter som enkapsler eller slår sammen attributtene til en eller flere entiteter. Det brukes i hovedsak for å begrense informasjon til sluttbrukere, slik at de ikke får tilgang til sensitive data og informasjon som kan kompromittere personvern og hovedmålet med plattformen som er anonymitet av søker.

3.4.2.3 Contracts

Contracts inneholder alle api-endepunktene som er tatt i bruk på Controllers og test prosjektene. DRY prinsippet²² er oppfylt ved å definere disse api-endepunktene i en fil. Dette reduserer redundant kode og hindrer endring og oppdatering av kode på flere filer.

²² [DRY](#) prinsippet står for Don't Repeat Yourself, som blir ofte brukt for å unngå repetisjon og redundante ved å introdusere abstraksjon.

```
public static class Auth
{
    private const string AuthControllerUrl = BaseUrl + "api/v1/auth";
    public const string Login = AuthControllerUrl + "/login";
    public const string Logout = AuthControllerUrl + "/logout";
}
```

Figur 3.4.2.3: Contract eksempel - Auth API endepunktene

3.4.3 Data Access Layer (DAL)

Data Access Layer er et lag som gir enklere tilgang til data lagret i databasen. Logikk som omhandler databasen blir implementert i dette laget adskilt fra controller, og på den måten oppnår vi en *Single Responsibility Prinsipp*, som vil si at hver del har sine egne ansvarsområder. I dette tilfelle har DAL kun ansvar for aksessering av database.

3.4.3.1 Models

Modellene representerer formen til dataene som er lagret i databasen. Dessuten, er modellene brukt som mal for oppbygging av dataoverføringsobjekter.

```
public class Admin
{
    [Key]
    public string Email { get; set; }
    public string Username { get; set; }
    public Role Role { get; set; }
    public byte[] HashedPassword { get; set; }
    public byte[] Salt { get; set; }
}
```

Figur 3.4.2.1:
Modell eksempel - Admin modell

3.4.3.2 Repositories

Repositorier er en abstraksjon av data aksess laget. Den skjuler detaljene på hvordan dataene er lagret eller hentet fra en datalagringsplass. Ved å implementere en *repository design pattern*, blir data aksess laget løst koblet, det vil si at hvis Entity Framework er erstattet med en annen ORM²³ som *Dapper* eller *Venflow*, implementasjonen på kontrollere blir ikke påvirket. Dette er fordi kontrollere bruker interfaçer. Disse interface-ene er en mal på operasjonene man må kunne utføre mot en datalagringsplass. Klasser er da opprettet og implementerer disse interface-ene. Man kan velge for eksempel å bruke Entity Framework til å implementere operasjonene som er definert i den. Hvis man ønsker å endre ORM-en til en annen, trenger man bare å oppdatere disse klassene. Dette vil ikke påvirke interface-ene og kontrollere.

3.5 Sikkerhet

En av ikke-funksjonelle kravene til systemet er sikkerhet. Sikkerheten på systemet skal sørge for å opprettholde integriteten, konfidensialiteten og autentisiteten. Med integritet og konfidensialitet menes det at ikke autoriserte parter ikke skal ha tilgang til data og ikke kan modifisere dataene. Mens autentisiteten går ut på å unngå forfalskning av data eller gjengivelse av dårlig data som kan kompromittere sikkerheten til systemet. Disse sikkerhetstiltakene er opprettholdt gjennom datavalidering, autorisering og autentisering av brukere.

²³ [ORM](#) står for Object-Relational Mapping

3.5.1 Datavalidering

Datavalidering brukes for å sikre autentisiteten ved bruk av regulære uttrykk²⁴ på både klient side og server siden. Den sørger for at all data som skrives av sluttbruker er begrenset til et mønster. Eksempelvis, ingen bokstaver i telefonnummer felt, lengde på passord som skal være på minst 8 tegn eller gyldig epostadresse.

3.5.2 Autentisering og autorisering

Autentisering er hvordan systemet identifiserer brukeren og autorisering skal bestemme hva brukeren har tilgang på i systemet. Disse to områdene av sikkerheten hjelper med å opprettholde konfidensialiteten og integriteten av systemet, og personvern av brukere.

Dette løses i systemet ved hjelp av Json Web Token (JWT) og C# .NET Authentication. JWT er en enkodet JSON objekt som består av tre deler; Header, Payload og Signatur. Header inneholder informasjon om hva slags signering algoritme er brukt f.eks. HMAC²⁵ SHA256²⁶. Andre delen Payload består av informasjon om brukeren som vil bli autorisert, f.eks. navn, epostadresse og rolle. I tillegg til det inneholder det når tokenet utløper. Siste delen av JWT er signaturen som brukes for å sjekke om tokenet er gyldig. Signaturen valideres på server, og sjekkes mot et privat nøkkel.

Tilslutt når tokenet er generert og signert av server brukes det som et Authorization Token på HTTP Header for hver forespørsel fra klienten. [23][24]

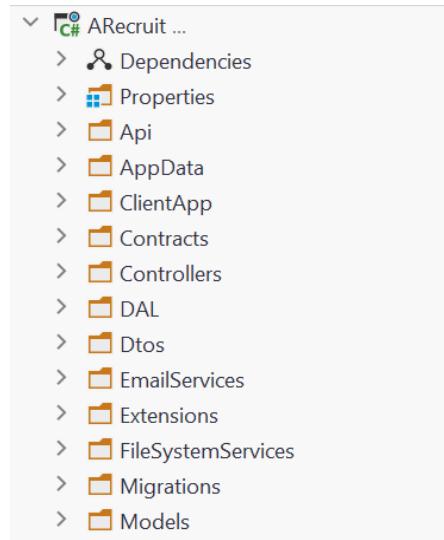
²⁴ [Regulære uttrykk](#) er en rekke tegn og karakterer som spesifiserer en søke mønster i tekster. [25]

²⁵ [HMAC](#) står for Hash-based Message Authentication Code. HMAC er en type message authentication code som bruker en hash-funksjon og en secret key for å generere den.

²⁶ [SHA256](#) er en type av Secure Hash Algorithms som består av 256 bit.

3.6 Filstruktur

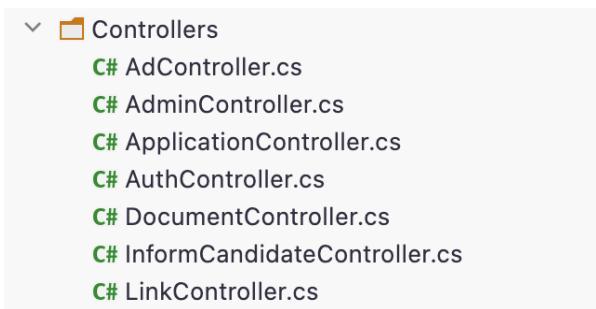
I denne seksjonen går vi kort gjennom prosjektets filstruktur. *Figur 3.6* viser en overordnet struktur av hele prosjektet.



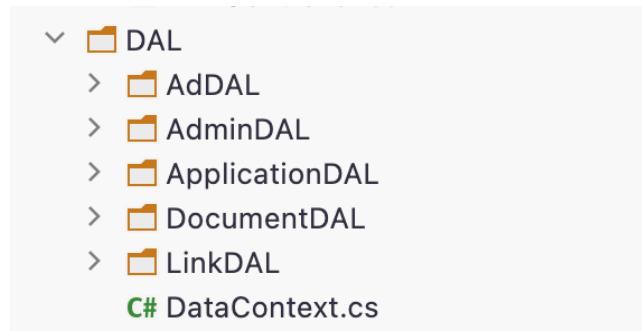
Figur 3.6: Prosjektets filstruktur

3.6.1 Backend

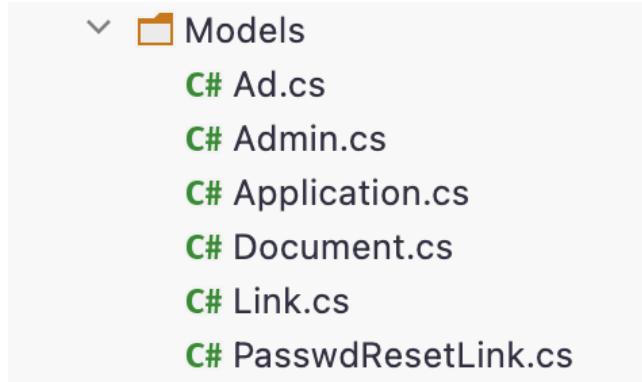
Backend filstrukturen består av *Controllers*, *DAL*, *Models*, *FileSystemServices*, *Extension* og *DTOS*.



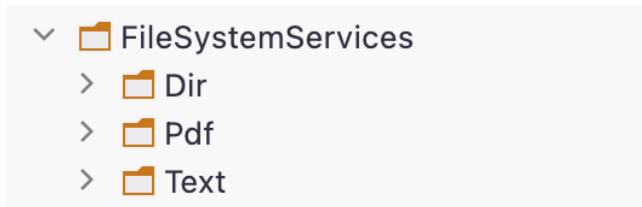
Figur 3.6.1-A: Controller (Les mer under [3.4.2.1 Controllers](#))



Figur 3.6.1-B: DAL - Data Access Layer ([les mer under 3.4.3 Data Access Layer](#))



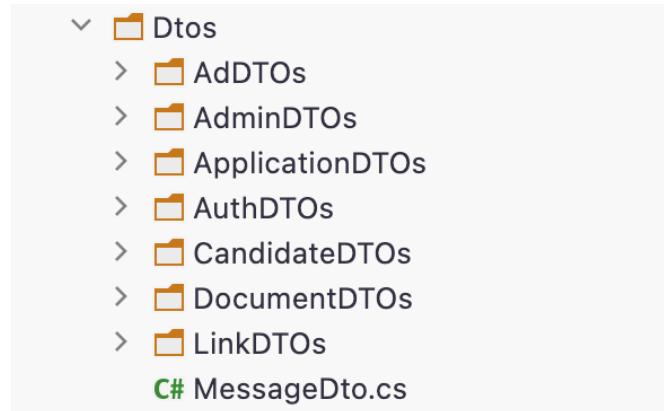
Figur 3.6.1-C: Models ([les mer under 3.4.3.1 Models](#))



Figur 3.6.1-D: FilsystemServices er mappe som inneholder relatert funksjonalitet til fillagring og -lesing fra filsystemet.



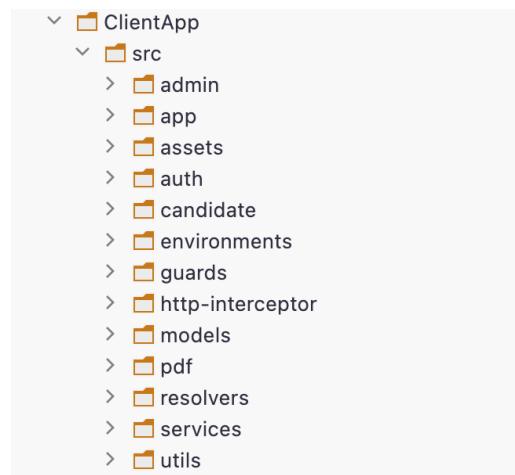
Figur 3.6.1-E: Extensions (utvidelser) er en mappe som inkluderer nødvendig implementasjon for generering av id, fil, JWT, passord sikring og mapping.



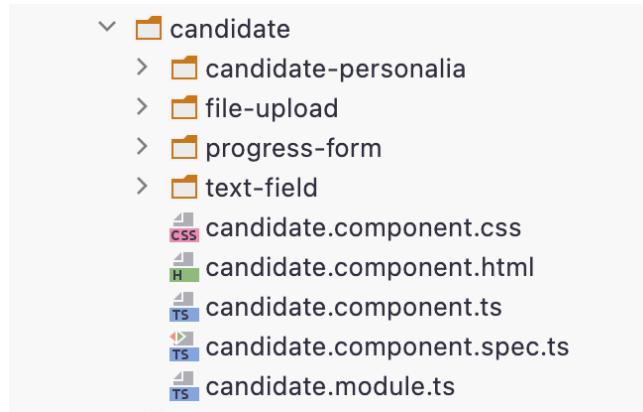
Figur 3.6.1-F: DTOs - Data Transfer Objects (*les mer under [3.4.2.2 Dtos](#)*)

3.6.2 Frontend

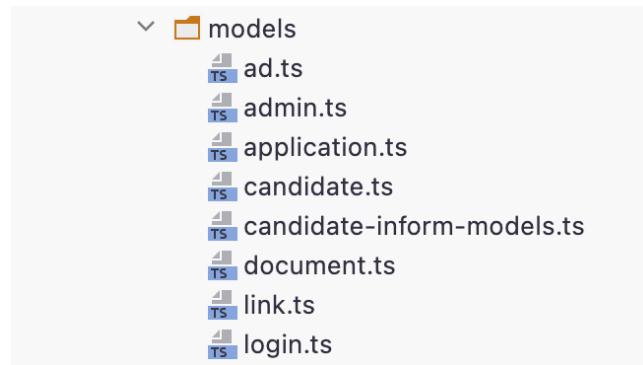
Frontend inkludere modules som inneholder componentes, services og models. *Figur 3.6.2-A viser et helhetlig bilde av ClientApp som er Frontend.*



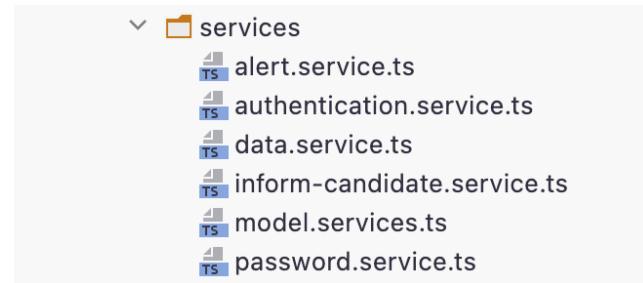
Figur 3.6.2-A: ClientApp filstruktur. Her kan vi se modulene **app**, **admin**, **auth**, **assets**, **candidate** og **utils**.



Figur 3.6.2-B: Eksempel på Components under candidate module. (*Les mer under [3.3.3 Komponenter](#)*)



Figur 3.6.2-C: Models ([Les mer under 3.3.4 Modeller](#))



Figur 3.6.2-D: Services ([Les mer under 3.3.6 Services](#))

3.7 Konklusjon

Produktet som er ferdig utviklet møter alle de kravspesifikasjoner som er ønsket fra oppdragsgiver. I tillegg til dem har gruppen lagt til et ekstra krav for systemet, slik at sluttbruker har mulighet til å nullstille passord.

Selv om produktet møter kravspesifikasjoner og er en fungerende applikasjon, har gruppen to forslag til videreutvikling av applikasjonen. Det første forslaget er en AI²⁷ basert anonymisering istedenfor en manual anonymisering av kandidater. Grunnen for det er et bedre personvern og en høyere anonymitetsgrad. Det andre forslaget er utvikling av en mobilapplikasjon for plattformen. Det er viktig, fordi i dagens samfunn har mobiltelefoner blitt en viktig del av hverdagen vår.

Utviklingsprosessen var ikke uten utfordringer. Gjennom prosessen har valget av tech stack og arkitektur skapt sine utfordringer. Først og fremst på grunn av gruppens kunnskapsmangel om å jobbe og utvikle i et stort prosjekt. Men disse utfordringene har vi klart å løse og ikke minst har disse utfordringene gitt oss gode muligheter til å lære av.

For å oppsummere, har utviklingen av produktet vært lærerikt for oss. Vi har følt på hvordan det er å utvikle et stort produkt for en kunde, og hvordan det er å jobbe med et stort prosjekt. Gjennom dette prosjektet har vi hatt muligheten til å jobbe med forskjellige teknologier, og løst ulike utfordringer gruppen har kommet bort i.

Vi har også kommet frem til, og reflektert at med store prosjekter så er det alltid rom for mer jobb og forbedringer, selv om produktet møter kravspesifikasjonene.

²⁷ [AI](#): Kunstig Intelligens

Kapittel 4

Testdokumentasjon

Innhold

- 4.1 Introduksjon**
- 4.2 Enhetstesting**
- 4.3 Integrasjonstesting**
- 4.4 Test oppsummering**

4.1 Introduksjon

Dette kapitlet inneholder en detaljert beskrivelse av testprosessen under utvikling av ARecruit plattformen. Testing av produktet er en egen fase som i vårt fremdriftsplan som heter testingsfase. Selv om i fremdriftsplanen er testingsfasen adskilt fra utviklingsfasen, men disse to fasene går fortsatt veldig tett i hverandre, spesielt i et smidig arbeidsmetodikk.

Testingsfasen består av enhetstesting og integrasjonstesting. Denne fasen er lagt til på planen for å kvalitetssikre applikasjonen. Hvor i enhetstesting testes det forretningslogikken, mens i integrasjonstesting testes sikkerhet, filsystem og database integrasjonen med systemet.

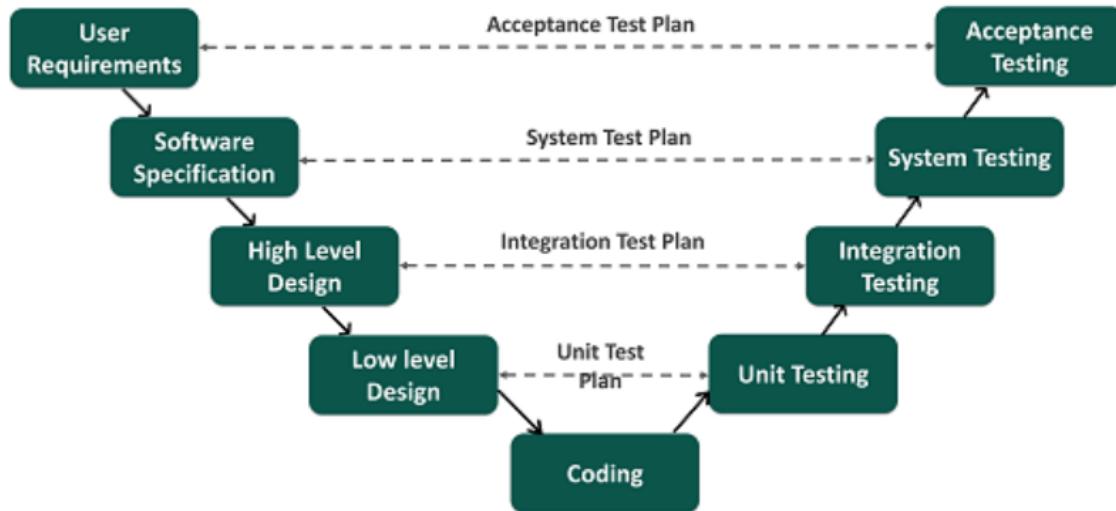
4.1.1 Testplan

ARecruit er en plattform som krever sikkerhet og god oppbevaring av kandidatenes informasjon. For en enkel feil i applikasjonens sikkerhet kan begå tilliten av plattformen, siden den påstår for å være en anonymisert rekrutteringsportal. Derfor var det viktig for oss å finne feil og å rette opp feilene allerede fra første steg.

Med hensyn på dette velger gruppen V-modell for testing av applikasjonen. V-modellen er en modell av programvare testing, hvor testene går fra laveste nivå til høyeste nivå av implementasjonen. Det vil si at testing begynner med å verifisere og validere fra forretningslogikken, og i senere steg skal det testes integrasjon mellom forskjellige systemer og til slutt testing av systemet som helhet. [26]

Gruppens plan for testingen inneholder først enhetstesting som tester laveste design nivå, som er forretningslogikken i Controller klassene. Deretter skal det gjennomføres

integrasjonstesting, for å validere og verifisere integrasjonen mellom databasen, filsystemet og API-et.



Figur 4.1.1: V-modell fra [27]

Utfra Figur 4.1.1 ser vi at systemspesifikasjoner testes med systemtesting, som er en god testingsmetode for å validere systemet eller produktet i helhet. Gruppen fikk ikke mulighet til å utføre systemtesting av ARecruit, på grunn av tranghet med tiden. Istedentfor systemtesting, testet alle medlemmene i gruppen produktet manuelt for å finne feil og ikke fungerende deler eller brukerhistorier som var ikke oppfylt.

4.1.2 Avvik i forhold til testplanen

Gruppen fikk til å utføre alle oppgavene som var planlagt for testing av produktet. Men testing hadde sine utfordringer som fikk arbeidet til å gå litt saktere enn forventet. Følgende er utfordringer som gruppen brukte mer tid på:

- Enhetstestene krevde uavhengighet og mock av tjenester. For denne grunn måtte vi implementere en del kode på nytt med Dependency Injection²⁸ mønsteret for å kunne mock-e for testene.
- Integrasjonstestene måtte ikke ha tilgang til hoveddatabasen og ekte tjenester som kunne ha bivirkninger. Derfor ble det brukt mye tid på å finne en god løsning for det.

4.2 Enhetstesting

Enhetstest er den første testmetoden i V-modellen, og starter helt tidlig i utviklingsfasen. Denne går ut på å avdekke alle mulige feil så tidlig som mulig, og tester individuelle metoder om de fungerer slik som ønsket. Alle mulige utfall av metodene skal testes med enhetstest, og da sjekkes det også om riktige feilmeldinger blir returnert om det oppstår feil.

4.2.1 Testobjekter

Det er alle metoder under kontrollene som skal testes en og en. Kontrollerne det gjelder er *AdController.cs*, *AdminController.cs*, *ApplicationController.cs*, *DocumentController.cs*, *LinkController.cs*, *InformCandidateController.cs* og *AuthController.cs*. Tilsammen er det 25 metoder som blir testet, og enhetstesten til disse blir utført av det medlemmet som implementerte metoden.

²⁸ [Dependency Injection](#) er en software design mønster hvor objekter får funksjonalitet ved å bli komponert av andre objekter som implementerer funksjonaliteten. I dette blir det komponerte objektet avhengig av implementasjonen. Det bidrar med en løs kobling mellom implementasjonen og det komponerte objektet.

4.2.2 Testmetodikk og testverktøy

Hvit-boks teknikken

Testene for enhetstesting ble implementert av oss selv gjennom utviklingsfasen.

Dermed hver oss visste hva vi skal teste, derfor kalles dette “*White Box Testing*”. Vi testet logikken ut fra vår kjennskap til logikken og metodene.

Selvstendige tester

En viktig del av enhetstesting var at alle testmetodene skulle være selvstendige og uavhengige av hverandre. Og ikke minst at hver testmetode skulle teste en enkelt tilfelle i logikken uavhengig av andre påvirkninger implementasjonen hadde. For eksempel en registreringsmetode har som oppgave å validere innkommende data til systemet. Og til slutt skal den lagre et nytt datapunkt i databasen. Derfor var det viktig for oss at testmetoden skulle bare teste valideringslogikken av data, men ikke lagringsprosessen på databasen.

For å oppnå uavhengige og selvstendige testmetoder, brukte vi avhengighetsinjeksjon (*Dependency Injection*) allerede fra starten på Controller implementasjonen. Senere i testingen hjelpt dette oss med å kunne Mock-e²⁹ disse avhengighetene ved hjelp av Moq³⁰.

²⁹ [Mock](#) er en prosess i enhetstesting som brukes når en komponent som testes har andre avhengigheter. I denne prosessen prøver testeren å forfalske resultater fra de avhengighetene.

³⁰ [Moq](#) er en bibliotek for å forfalske funksjonalitet.

Automatiseringsverktøy og skrive stilen

En av kravene på testing er å kunne repetere dem. For å oppfylle dette kravet gruppen automatiserte tester for enhetstesting. Disse testene ble implementert i et prosjekt som bruker NUnit³¹ biblioteket som er bygt på C# språket.

I tillegg til automatisering av tester måtte gruppen skrive testene på en ryddig måte slik de er lett å forstå for videre utvikling. Derfor brukte vi trippel-A skrivestilen.

Trippel-A står for AAA som er Arrange, Act og Assert. I Arrange delen skal det settes opp alt nødvendig for at metoden skal testes, disse nødvendige tingene kan være dataobjekter og mock av metoder og deres returverdi. Etterpå kommer Act hvor det skal gjøres et kall på metoden som skal testes, ved bruk av verdiene i Arrange som argumenter. Til slutt skal resultatet fra Act blir validert med Assert.

Figur 4.2.2 viser et kodeeksempl på skrivestilen:

```
[Test]
public async Task CreateAdFailsWhenAdminNotFound()
{
    //arrange
    _repository.Setup(repo
        => repo.Create(It.IsAny<string>(), It.IsAny<Ad>()))
        .ReturnsAsync((Ad) null);
    _mapper.Setup(m => m.Map<Ad>(adDto)).Returns(ad);

    //act
    var result = await _controller.Create(adLinkDto);
    var actionResult = result as ObjectResult;

    //assert
    Assert.AreEqual(StatusCodes.Status400BadRequest, actionResult.StatusCode);
    Assert.AreEqual(new MessageDto(){Message=$"Admin med epost: {adLinkDto.Ad.Email} finnes ikke!"}.ToString(),
        actionResult.Value.ToString());
}
```

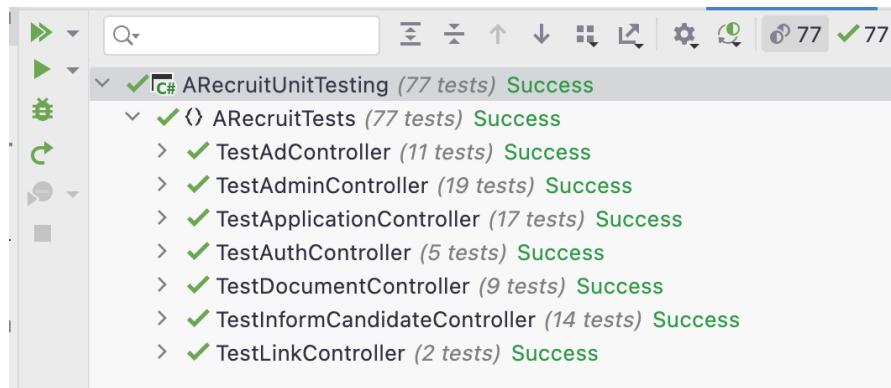
Figur 4.2.2: Testmetode

³¹ [NUnit](#) er en testbibliotek for C#. [28]

4.2.3 Konklusjon av enhetstesting

4.2.3.1 Resultater

Figur 4.2.3.1-A viser resultatene av enhetstesting. Det ble tilsammen 77 enhetstester, og alle ble bestått uten feil. Figur 4.2.3.1-B viser også code coverage³² av testene. Code coverage er et mål på hvor mye av koden som er dekket av testene, og skal dermed sikre at vi har testet alle linjer i kontrollene, med andre ord forretningslogikken. Code coverage for kontrollene er 100%.



Figur 4.2.3.1-A: Enhetstest resultater



Figur 4.2.3.1-B: Code Coverage

³² [Code Coverage](#) er en måleenhet i testing som avgjør hvor mange linjer av koden har gått gjennom testing.

4.2.3.2 Konklusjon

Enhetstesting spiller en viktig rolle i programvareutvikling. Enhetstesting forbedrer kodekvaliteten betydelig. Det hjelper utviklere med å identifisere de minste feilene som kan være tilstede i koden før de går til integrasjonstesting. Enhetstesting lar programutviklere å tenke gjennom utformingen av programvaren og hva som må gjøres før de implementerer koden. Dette fører til økt effektivitet, redusert kostnader og redusert nedetid av systemet.

4.3 Integrasjonstesting

Til forskjell fra enhetstest som tester en og en metoder hver for seg, skal integrasjonstest teste sammenhengen mellom flere moduler og interaksjonene mellom de, fra API grensesnittet helt ned til databasen. Poenget med denne testen er å sikre at alle modulene og metodene som blir kalt er kompatible med hverandre og fungerer som ønsket.

4.3.1 Testobjekter

Testobjektene for integrasjonstesting er alle API-endepunktene som er følgende:

Entitets Navn	API-endepunkter
Ad (Stilling)	<ul style="list-style-type: none">- ad/- ad/{id:int}- ad/create- update/{id:int}&ns={isActive:bool}
Admin	<ul style="list-style-type: none">- admin/- admin/create- admin/update/{email}- admin/resetPassword/email={email}&id={id:guid}

	<ul style="list-style-type: none"> - admin/delete/{email} - admin/sendPasswordResetLink/email={email} - admin/passwordResetLink/{id:guid}
Application	<ul style="list-style-type: none"> - application/create/{adId:int} - application/{adId:int} - application/status/{id:int} - application/candidate/{id:int}
Document	<ul style="list-style-type: none"> - document/{id:int} - document/{id:int}/anonymous - document/update/{id:int}
Link	<ul style="list-style-type: none"> - link/get/{adId:int}
Auth	<ul style="list-style-type: none"> - api/v1/auth/login
InformCandidate	<ul style="list-style-type: none"> - informCandidate/invitation - informCandidate/rejection - informCandidate/confirmation

4.3.2 Testmetodikk og testverktøy

Hvit-boks teknikk

Integrasjonstesting er utført ved bruk av hvit-boks teknikken. Gruppen har spesielt fokusert på å teste api-endepunktene som vil hente og gi riktig data fra database til klienten. Testingen krever at en tester har intern kunnskap om applikasjonen, dermed er hvit-boks teknikken mer egnet å bruke enn svart-boks teknikken. Ved gjennomføring av denne testingen, har gruppen fått innblikk på hvordan serveren og klienten kommuniserer og overfører data med hverandre.

Automatiseringsverktøy og skrive stilen

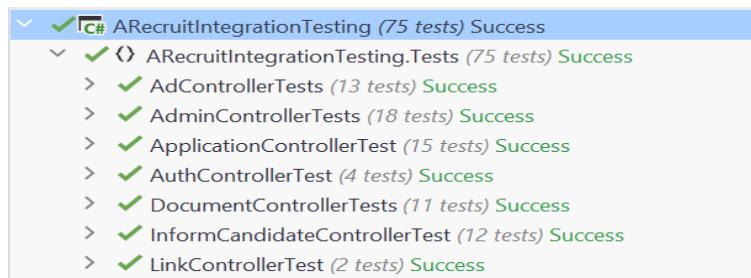
For automatisering av testene brukte vi en xUnit prosjekt som er en bibliotek for testing i C#. Men for at testene skulle være egnet til repetisjon måtte gruppen lage en In Memory database som blir gjenopprettet for hver test. I tillegg til det, måtte vi kjøre testene slik at de ikke påvirker data og andre tjenester på systemet. For å løse dette dette problemet implementerte vi en del kode for å opprette In Memory klienter som gjør HTTP spørninger uten å påvirke selve prosjektet.

Implementeringen av integrasjonstestene følger AAA (Arrange, Act, Assert) mønsteret for å ordne og formattere kode. Disse metodene tester kun logikken som involverer kommunikasjonen mellom serveren og klienten, det vil si, om riktig http protokoller og data er utvekslet mellom de to.

4.3.3 Konklusjon av integrasjonstesting

4.3.3.1 Resultater

Figur 4.3.3.1 viser alle resultatene for integrasjonstesten. Det er 75 tester og alle testene ble bestått.



Figur 4.3.3.1: Resultater integrasjonstest

4.3.3.2 Utfordringer og læringsutbytte

En krevende forarbeid for integrasjonstesting var det å konfigurere en database med riktig data og en klient som skulle ikke bruke de reelle tjenestene som var implementert i applikasjonen. Det å utfylle dette kravet var en utfordring for gruppen. Det krevde mye av oss siden vi hadde ikke erfaring med dette. Etter mye lesning og søk etter løsning fra forskjellige dokumentasjoner og artikler som blant annet fra Microsoft, til slutt klarte vi å løse dette.

Løsningen ble å lage falsk data og falske klienter som kjørte versjoner av API-et i lokalt i minnet, men ikke over nettverket. Vi klarte det ved hjelp av en klasse fra *Microsoft.AspNetCore.Mvc.Testing*. Klassen heter *WebApplicationFactory* som lar oss å kjøre en duplikat av API i minnet.

Til oppsummering av prosessen kan si at det var en utfordrende oppgave, men samtidig var den veldig lærerikt.

```
namespace ARecruitIntegrationTesting
{
    [ 11 usages  NimaAB  1 exposing API]
    public class ARrecruitWebApplicationFactory
        : WebApplicationFactory<Startup>
    {
        [ 3 usages]
        public IConfiguration Configuration { get; private set; }
        [ NimaAB]
        protected override void ConfigureWebHost(IWebHostBuilder builder)
        {
            builder.ConfigureAppConfiguration(config:IConfigurationBuilder =>
            {
                Configuration = new ConfigurationBuilder()
                    .AddJsonFile("appsettings.IntegrationTests.json") // IConfigurationBuilder
                    .Build(); // IConfigurationRoot

                config.AddConfiguration(Configuration);
            });

            builder.ConfigureServices(services:IServiceCollection =>
            {
                var descriptor = services.SingleOrDefault(
                    d:ServiceDescriptor => d.ServiceType ==
                    typeof(DbContextOptions<DataContext>));

                services.Remove(descriptor);

                services.AddDbContext<DataContext>(optionsAction:options =>
                {
                    options.UseInMemoryDatabase(databaseName: "InMemoryDbForTesting");
                });

                var sp:ServiceProvider = services.BuildServiceProvider();

                using (var scope = sp.CreateScope())
                {
                    var scopedServices:ServiceProvider = scope.ServiceProvider;
                    var db = scopedServices.GetRequiredService<DataContext>();
                    DataInitializer.InitializeForTest(db,
                        Configuration.GetSection(key: "AppSettings:datasource").Value);
                }
            });
        }
    }
}
```

Figur 4.3.3.2: Bildet viser koden som er brukt for å kunne kjøre API-et i minnet, og oppretting av ny database.

4.3.3.3 Konklusjon

Integrasjonstesting er det andre trinnet i hierarkiet av testnivåer. Hovedmålet med integrasjonstesting er å sikre tilkoblingen mellom individuelle moduler i et system.

Integrasjonstesting sørger for synkroniteten og kompatibiliteten av moduler, komponenter eller systemer med hverandre.

4.4 Test oppsummering

Ser vi tilbake på alt arbeidet som er utført på prosjektet i forhold til testing, kan vi si at det er en suksess. Gruppen klarte ikke bare å dokumentere, implementere og kjøre flere testcaser på to test nivåer, men også erkjenne behovet og viktigheten av å komme opp med løsninger på problemer som kommer med nye implementasjoner og endringer i koden.

Kapittel 5

Brukermanual

Innhold

- 5.1 Logg inn side**
- 5.2 Inspektør**
- 5.3 Beslutningstaker**
- 5.4 Jobbsøking for kandidater**
- 5.5 Andre komponenter**

5.1 Logg inn side

Denne seksjonen viser en detaljert steg for steg manual for logg inn siden til ARecruit applikasjonen.

5.1.1 Innlogging for administrasjon brukere

1. Gå til logg inn siden til ARecruit, og skriv inn e-postadressen og passordet.
2. Trykk på “Logg inn” knappen, eller enter knappen på tastaturet.
3. Hvis input verdiene samsvarer med det som er lagret i databasen, vil systemet videresende deg til din spesifikke admin side etter hvilken rolle du er registrert med.

Din bedrifts rekruteringsportal

The screenshot shows a dark blue login form. At the top, there is a header with the text "Din bedrifts rekruteringsportal". Below the header, there are two input fields: "E-post adresse" containing "hans@bedrift.com" and "Passord" containing ".....". Below the input fields is a large blue button labeled "Logg inn". At the bottom right of the form, there is a link "Glemt passord?".

5.1.2 Tilbakestilling av passord

1. Når du trykker på “Glemt passord?”, blir du sendt til send e-post siden.

A screenshot of a web page with a dark blue background. In the center, there is a white input field labeled "Hva er din epost adresse?" followed by a "Send" button. Below the button is a link "Innlogging siden".

2. Fyll inn e-post feltet og trykk på send. Hvis e-posten er gyldig, blir du send en epost som inneholder en lenke på denne formen: (NB: e-post innholdet finner du i Konsollen der programmet kjøres fra.)
<http://localhost:5000/auth/resetPassword/8c590c17-898e-4e1a-b350-6e408b4d9a69>
3. Ved å åpne lenken blir du sendt til en passord nullstilling form.
4. Skriv inn e-postadressen og lag et nytt passord.

A screenshot of a web page with a dark blue background. It shows three input fields: "Epost" containing "hans@hansen.com", "Ny passord" containing ".....", and "Bekreft passordet" containing "....." with an eye icon to its right. Below these fields is a "Send" button and a link "Innlogging siden".

5. Hvis epost og passord er riktig blir dermed passordet nullstilt for deg.

5.2 Inspektør

Denne seksjonen viser en detaljert steg for steg manual for inspektør.

5.2.1 Anonymisering av kandidater

1. Inspektør logger seg inn med sin e-postadresse og passord (se [5.1.1 Innlogging for administrasjon brukere](#)).
2. Trykk på den ønskede stillingen. Du vil da bli ført til hovedsiden til stillingen med oversikt over søker.



3. Ved å trykke på en søker vil du bli sendt til en personalia side av søkeren og du får tilgang til CV- og søknaden. Nye søker, som ikke er påbegynt anonymisering blir kategorisert under "Nye, ikke vurderte kandidater".

Nye, Ikke vurderte kandidater

Ola Norman
himmelsk_tyttebær
Ikke anonymisert

Alle kandidater

Alle Godkjente Avviste Avventer

- Trykk på en fil du ønsker å anonymisere for å bli videreført til anonymiseringssiden.

Deltidsjobb

Se annonsen

Personalia

Anonym ID:
himmelsk_tyttebær

Navn:
Ola Norman

Telefonnummer:
12345678

E-post:
ola@norman.com

CV:
 [CV_himmelsk_tyttebær.pdf](#)

Søknad:
 [Søknad_himmelsk_tyttebær.pdf](#)

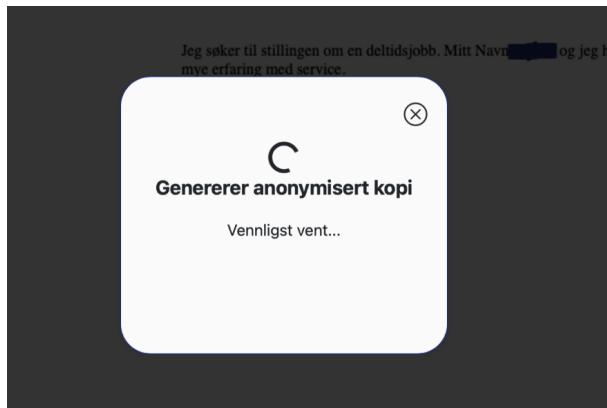
Tilbake

- Her kan du nå bruke sensureringsverktøyet for å starte å anonymisere søkeren.

Bruk "Sensurer" knappen for å markere over personlige opplysninger, "Angre" knappen for å angre en markering, og "Tilbake" knappen for å gå tilbake til personalia siden



6. Når du er ferdig med å sensurere filen, trykker du på "Lagre endringer" for å lage en anonymisert kopi av filen, og trykk tilbake knappen igjen for å gå tilbake til personalia siden til kandidaten.



7. Du vil nå se på personaliasiden at en anonymisert fil av originalen har blitt laget. Gjenta samme stegene med neste fil.

Deltidsjobb

[Se annonsen](#)

Personalia

Anonym ID:

himmelsk_tyttebær

Navn:

Ola Norman

Telefonnummer:

12345678

E-post:

ola@norman.com

CV:

[CV_himmelsk_tyttebær.pdf](#)

Søknad:

[Søknad_himmelsk_tyttebær.pdf](#)

[Søknad_himmelsk_tyttebær_anonym.pdf](#)

[Tilbake](#)

- Når du har ferdig anonymisert begge filene, går du tilbake til personalia siden og trykker på "Ja" på boksen under. Denne kommer opp bare når det er generert en anonym kopi av hver av filene. Deretter bekrefter du igjen at kandidaten er ferdig anonymisert.

CV:

[CV_himmelsk_tyttebær.pdf](#)

[CV_himmelsk_tyttebær_anonym.pdf](#)

Søknad:

[Søknad_himmelsk_tyttebær.pdf](#)

[Søknad_himmelsk_tyttebær_anonym.pdf](#)

Er kandidaten ferdig anonymisert?

[Ja](#)

[Tilbake](#)

9. Nå har søknaden til kandidaten blitt sendt videre til beslutningstaker for vurdering, og kandidaten får statusen “Venter på godkjenning”

Nye, Ikke vurderte kandidater

Alle kandidater

[Alle](#) [Godkjente](#) [Avviste](#) [Avventer](#)

himmelsk_tyttebær
Venter på godkjenning

5.2.2 Kontakt kandidat

1. Når en kandidat har blitt ferdig vurdert av beslutningstaker og fått status “Godkjent”, kan du tilkalle kandidaten til intervju.

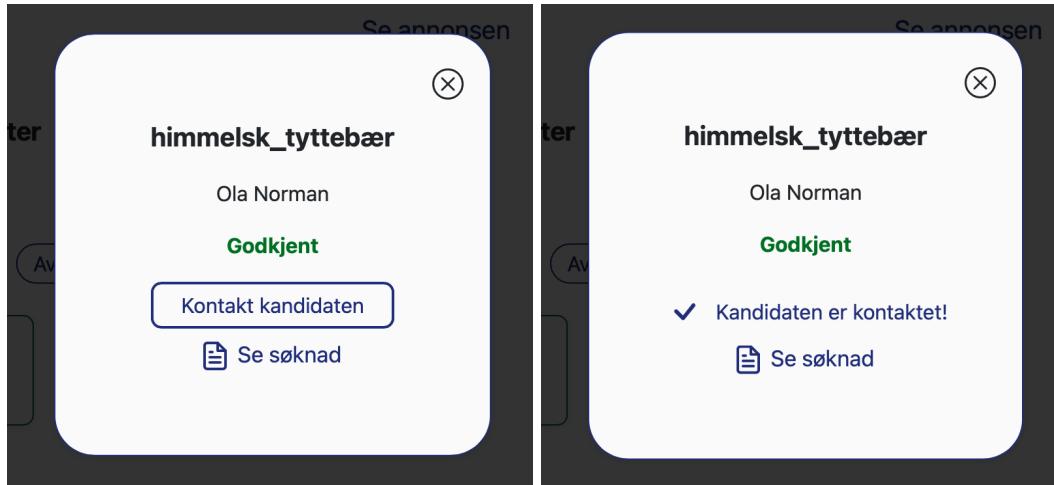
Nye, Ikke vurderte kandidater

Alle kandidater

[Alle](#) [Godkjente](#) [Avviste](#) [Avventer](#)

himmelsk_tyttebær
Godkjent

2. Når du trykker på kandidaten vil få opp en boks hvor du kan se informasjon om kandidaten, med en “Kontakt kandidaten” knapp som kan brukes for å tilkalle til intervju.



3. Dermed vil en mail bli sendt til kandidaten om kandidaten er godkjent, og samme når kandidaten blir ikke godkjent.
4. Samme prosedyre gjelder for ikke godkjente kandidater.

5.3 Beslutningstaker

Denne seksjonen viser en detaljert steg for steg manual for beslutningstaker.

5.3.1 Vurdering av kandidater

1. Beslutningstaker logger seg inn med e-postadresse og passord (se [5.1.1 innlogging for administrasjon brukere](#)), du vil da komme inn på siden med oversikt over stillinger.
2. Får å se kandidater til en bestemt stilling, trykker du på den ønskede stillingen.

Din bedriftens rekrutteringsportal

Aktive stillingsannonser

Deltidsjobb

 Legg til stillingsannonse

3. Du vil nå kun se kandidater som har blitt ferdig anonymisert av en inspektør.

Kandidater som har søkt, men ikke blitt anonymisert vil dermed ikke bli vist hos en beslutningstaker.

Alle kandidater

Alle

Godkjente

Avviste

Avventer

himmelsk_tyttebær

Venter på godkjenning

4. Trykk på kandidaten med status “Venter på godkjenning” for å vurdere. Du vil da bli ført til en oversiktsside hvor du kan se både CV og søkerbrevet til kandidaten som har blitt anonymisert. Trykk på filene for å lese innholdet.

himmelsk_tyttebær



[CV_himmelsk_tyttebær_anonym.pdf](#)



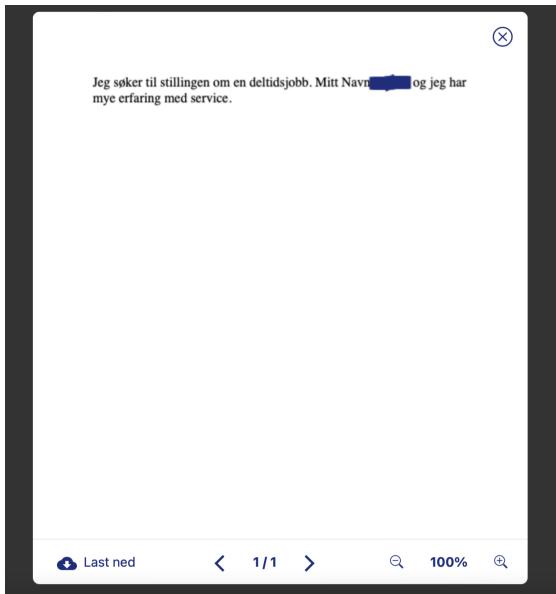
[Søknadsbrev](#)

[Søknad_himmelsk_tyttebær_anonym.pdf](#)

[Tilkal til intervju](#)

[Avslå søknad](#)

5. Her kan du lese søkerdokumentet og CV-en, du får muligheten til å laste ned filen du leser ved å trykke på knappen “Last ned” under filen, kan bla mellom sidene og forstørre eller minske filen.



6. Når du har tatt en avgjørelse av kandidaten kan du nå velge å enten tilkalle kandidaten til et intervju, eller avslå søknaden, og statusen til kandidaten blir endret etter avgjørelsen om det er godkjent eller ikke godkjent.

Tilkal til intervju

Avslå søknad

Alle kandidater

Aller Godkjente Avviste Avventer

himmelsk_tyttebær
Godkjent

Alle kandidater

Aller Godkjente Avviste Avventer

himmelsk_tyttebær
Ikke godkjent

7. Avgjørelsen blir nå sendt tilbake til inspektør som har ansvaret for å kontakte kandidaten.

5.3.2 Legg til nye stillinger

1. For å opprette nye stillinger trykker du på “Legg til stillingsannonse” på siden med oversikt over stillinger.

Din bedriftens rekrutteringsportal

Aktive stillingsannonser

Deltidsjobb

 Legg til stillingsannonse

2. Du vil få opp et skjema. Fyll ut skjemaet med navn på stillingen, og knytt lenke til annonsen.

Legg til ny stillingsannonse

Navn på stillingsannonsen

Fullstack utvikler søkes

Link til søkeradsskjema

lenken blir generert automatisk! 

Link til stillingsannonsen

<https://www.finn.no/job/fulltime/ad.html?finnkode=256892333>

 Legg til ny link

3. For å legge til flere lenker, trykk på “Legg til ny link”, du vil da få enda ett felt for å skrive en lenke. Trykk på “fjern” om du ikke ønsker å fylle den ut.

Legg til ny stillingsannonse

Navn på stillingsannonsen

Fullstack utvikler søkeres

Link til søknadsskjema

lenken blir generert automatisk!



Link til stillingsannonsen

<https://www.finn.no/job/fulltime/ad.html?finnkode=256892333>

Link til stillingsannonsen

Internship Siemens



Dette feltet er obligatorisk

Legg til ny link

4. Når hele skjemaet er fylt ut kan du trykke på knappen opprett lengst til høyre for å opprette stillingsannonsen. Feltet “link til stillingsannonsen” vil nå bli oppdatert med en lenke til søknadsskjemaet søker kan bruke for å søke til stillingen.

Link til søknadsskjema

<http://localhost:5000/apply-for/ad/7>



5.4 Jobbsøking for kandidater

Denne seksjonen viser fremgangsmåten for å sende søknad.

1. Du finner lenken fra en jobbannonse nettside.
2. Først blir du sendt til personalia formen for å fylle ut dine personlige informasjoner.
3. Hvis personalia er riktig, så blir “Neste” knapp aktivert og den kan trykkes på for å gå til neste side.

deltidsjobb

Se annonen



Personalia



CV



Søknadsbrev



Fullfør

Hva heter du?

Nima Abdollahi

E-post adresse

nima@abdollahi.com

Telefonnummer

40033425

Fødselsdato

25.05.1998

Neste

4. Her skal det lastes opp en CV dokument i pdf format. Du kan laste opp dokumentet ved "drag og drop" eller ved å trykke på "Last opp PDF". "Slett fil" knappen kan brukes for å slette filen som er lastet opp. Ved å trykke på "Neste" knapp går du videre og med "Tilbake" går du til forrige side.

The image displays two side-by-side screenshots of a digital application form interface. Both screenshots show the top navigation bar with four tabs: 'Personalia', 'CV' (which is highlighted in grey), 'Søknadsbrev' (Cover Letter), and 'Fullfør' (Finish). Below the tabs is a large input field with a dashed border, containing a cloud icon with an upward arrow and the text 'Drag&Drop PDF her'. Below this, the word 'eller' (or) is centered, followed by a blue-outlined button labeled 'Last opp PDF'. A pink rectangular box at the bottom of the field contains the text 'Dette feltet er obligatorisk' (This field is mandatory). At the very bottom of the left screenshot, there are two buttons: 'Tilbake' (Back) and 'Neste' (Next). In the right screenshot, the same input field is shown, but it now contains the text 'CV-Nima.pdf' next to the cloud icon. Below this text is a blue-outlined button labeled 'slett fil' (Delete file). The bottom buttons 'Tilbake' and 'Neste' are also present.

5. På denne siden skal det lastes opp søkerbrev enten som en pdf-fil eller så kan du skrive en søkerstekst i tekstufllet. Ved å trykke på "Neste" går du til siste siden av søkerformen.



A dashed rectangular area containing the following elements:

- A blue cloud icon with an upward arrow.
- The text "Drag&Drop PDF her".
- The word "eller" (or).
- A blue button labeled "Last opp PDF".

Below this area, there is a file preview section:

- A small blue document icon followed by the text "sokand.pdf".
- A blue button labeled "slett fil" (Delete file).

eller

Skriv søknadsbrev

A large, empty rectangular text input field with a cursor at the bottom right corner.

Tilbake Neste

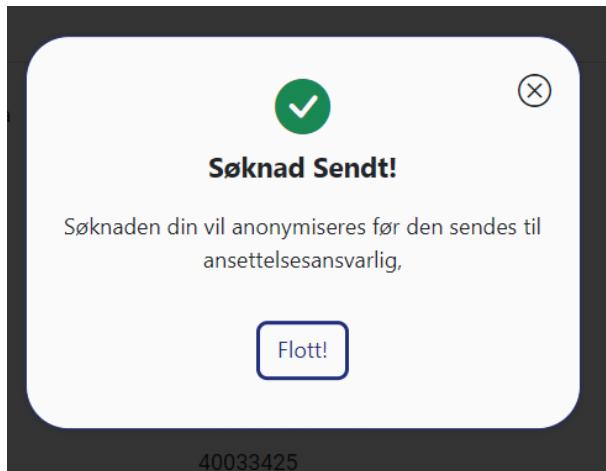
6. Her på siste side vises en oppsummering av informasjonen som er fylt opp.

The screenshot shows the 'deltidsjobb' application interface. At the top, there is a blue header bar with the text '5 Brukermanual'. Below the header, the title 'deltidsjobb' is displayed in a large, bold, dark blue font. Underneath the title is a button labeled 'Se annonsen'. Below this, there is a horizontal navigation bar with four items: 'Personalia' (with a person icon), 'CV' (with a document icon), 'Søknadsbrev' (with a document icon), and 'Fullfør' (with a checkmark icon). The 'Personalia' item is currently selected. To the right of the navigation bar, there is a link 'Se annonsen'. The main content area is titled 'Personalia' and contains the following information:

- Anonym ID: (empty)
- Navn: Nima Abdollahi
- Telefonnummer: 40033425
- E-post: nima@abdollahi.com
- CV: [CV-Nima.pdf](#)
- Søknad: [soknad.pdf](#)

At the bottom of the content area are two buttons: 'Tilbake' and 'Send'.

7. Ved å trykke på send blir søknaden sendt videre til behandling til inspektør brukere.

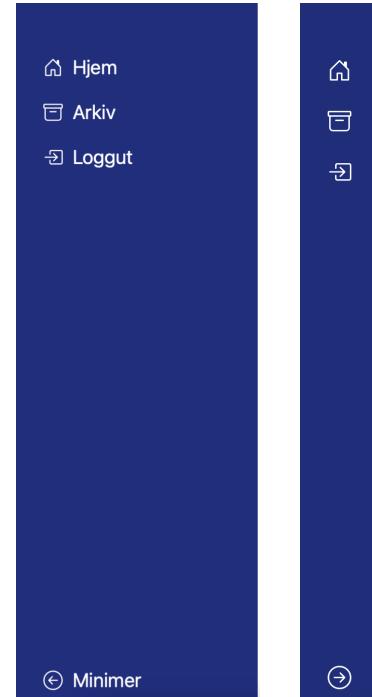


8. Med det blir det sendt en epost som blir skrevet på konsollen der programmet kjøres fra.

5.5 Andre komponenter

5.5.1 Navigasjonsmenyen

1. Trykke på “Hjem” for å vise aktive stillinger.
2. Trykke på “Arkiv” for å vise arkiverte stillinger.
3. Trykk på “Loggut” for å bli logget ut av siden.
4. Trykk på “Minimer” for å minimere sidenavigasjonen.
5. Trykk på pilen igjen for å forstørre menyen.



5.5.2 Filtrere kandidater

- For å kun se godkjente kandidater, trykk på “Godkjente” i menyen over kandidatlisten

Alle kandidater

Aller Godkjente Avviste Avventer

himmelst_tyttebær
Godkjent

glada_hvitløk
Godkjent

- For å kun se kandidater som enda ikke har blitt vurdert, trykk på “Avventer”

Alle kandidater

Aller Godkjente Avviste Avventer

djerv_sukkerrør
Venter på godkjennning

- For å kun se avviste kandidater, trykk på “Avviste”.

Alle kandidater

Aller Godkjente Avviste Avventer

bestemt_bukko
Ikke godkjent

Ordliste

API er en forkortelse for Application Programming Interface som er en mellommann som kobler flere deler av applikasjoner eller systemer sammen.

Database er en organisert kolleksjon av strukturert data og informasjon.

Sprint er en iterasjon av utviklingsprosess i softwareutvikling, det varer i 1-2 uker som vanlig. [\[11\]](#)

Sprint backlog er en liste av planen som skal utføres i sprinten. [\[11\]](#)

ASP .NET Core er en rammeverk produsert av Microsoft for C#, F# og VisualBasic språkene. Rammeverket blir brukt for utvikling av webapplikasjoner. [\[12\]](#)

SQLite er en database engine som er basert på SQL. [\[13\]](#)

EntityFramework er en database rammeverk fra Microsoft, som gjør det enkelt å integrere database med applikasjonen. [\[14\]](#)

DAL er det laget i applikasjonens design arkitektur som har tilgang til databasen.

JWT står for Json Web Token. Det er en objekt som inneholder litt enkodet data som brukes for å autentisere brukere.

UserClaims er enkodet objekter som blir bevart i JWT objektet og senere blir de brukt for, autentisering og autorisering av brukeren.

xUnit.net er en bibliotek i C# språket som brukes for automatisering av tester. [\[15\]](#)

Angular er en TypeScript rammeverk for utvikling av webapplikasjoner utviklet av Google.

NPM (Node Package Manager) står for Node pakke behandler, det hjelper med nedlasting og organisering av nødvendige pakker. [\[16\]](#)

UML står for Unified Modeling Language som blir ofte bruk for presentering av en software før den skal utvikles.

TypeScript er et objektorientert programmeringsspråk.

Bootstrap er en enkelt og raskt rammeverk for design og oppbygning av nettsider.

HTTP står for Hypertext Transfer Protocol er en protokoll i applikasjonslaget i internett protokollen.

CRUD operasjonene er forkortelse for operasjonene opprett, les, endre og slett.

MVC står for Model, View, Controller

Bearer Token er en HTTP ordning som brukes mest for sikkerhet av rutene.

JSON er Javascript objektnotasjon som er en fil og data format. Det blir ofte brukt i veksling av data mellom flere systemer. [\[22\]](#)

DRY prinsippet står for Don't Repeat Yourself, som blir ofte brukt for å unngå repetisjon og redundante ved å introdusere abstraksjon.

ORM står for Object-Relational Mapping.

Regulære uttrykk er en rekke tegn og karakterer som spesifiserer en søke møster i tekster. [\[25\]](#)

HMAC står for Hash-based Message Authentication Code. HMAC er en type message authentication code som bruker en hash-funksjon og en secret key for å generere den.

SHA256 er en type av Secure Hash Algorithms som består av 256 bit.

AI: Kunstig Intelligens

Dependency Injection er en software design mønster hvor objekter får funksjonalitet ved å bli komponert av andre objekter som implementerer funksjonaliteten. I dette blir det komponerte objektet avhengig av implementasjonen. Det bidrar med en løs kobling mellom implementasjonen og det komponerte objektet.

Mock er en prosess i enhetstesting som brukes når en komponent som testes har andre avhengigheter. I denne prosessen prøver testeren å forfalske resultater fra de avhengighetene.

Moq er en bibliotek for å forfalske funksjonalitet.

NUnit er en testbibliotek for C#. [\[28\]](#)

Code Coverage er en måleenhet i testing som avgjør hvor mange linjer av koden har gått gjennom testing.

Bibliografi

- [1] Magnus, K. (2021, June 22). *WAYS innfører anonymisert rekruttering - WAYS*. WAYS AS. Retrieved May 16, 2022, from
<https://ways.no/innsikt/ways-innfoerer-anonymisert-rekruttering>
- [2] Zoom. (n.d.). *About*. Zoom. Retrieved May 16, 2022, from
<https://explore.zoom.us/en/about/>
- [3] Google. (n.d.). *How to Use Google Meet Video Conferencing | Google Meet*. Google Apps. Retrieved May 16, 2022, from
<https://apps.google.com/meet/how-it-works/>
- [4] Git. (n.d.). *Git*. Git SCM. Retrieved May 16, 2022, from
<https://git-scm.com>
- [5] Atlassian. (n.d.). *A brief overview of Jira*. Atlassian. Retrieved 05 18, 2022, from
<https://www.atlassian.com/software/jira/guides/getting-started/overview#jira-software-hosting-options>
- [6] Figma. (n.d.). *Design, prototype, and gather feedback all in one place with Figma*. Figma. Retrieved May 18, 2022, from
<https://www.figma.com/design/>
- [7] Bullock, D. (n.d.). *Google Sheets: Online Spreadsheets for Business*. Google Workspace. Retrieved May 18, 2022, from
<https://workspace.google.com/products/sheets/>

- [8] Wikipedia. (n.d.). *Diagrams.net*. Wikipedia. Retrieved May 16, 2022, from <https://en.wikipedia.org/wiki/Diagrams.net>
- [9] John, S. (2021, February 18). *What Is Slack? the Professional Messaging Program, Explained*. Business Insider. Retrieved May 18, 2022, from <https://www.businessinsider.com/what-is-slack?r=US&IR=T>
- [10] Kristoffersen, B. (2020). Normalisering. In *DATABASESYSTEMER*. (5th ed., pp. 235-242). UNIVERSITETSFORLAGET. 978-82-15-03251-1
- [11] Sommerville, I. (2011). Agile project management. In *Software Engineering* (9th ed., pp. 72-73). Pearson. 978-0-13-705346-9
- [12] Microsoft. (2022, May 12). *ASP.NET overview*. Microsoft Docs. Retrieved May 18, 2022, from <https://docs.microsoft.com/en-us/aspnet/overview>
- [13] SQLite Home Page. (n.d.). SQLite Home Page. Retrieved May 16, 2022, from <https://www.sqlite.org/index.html>
- [14] Microsoft. (2021, July 14). *Compare EF6 and EF Core*. Microsoft Docs. Retrieved May 16, 2022, from <https://docs.microsoft.com/en-us/ef/efcore-and-ef6/>
- [15] Xunit. (n.d.). Home > xUnit.net. Retrieved May 16, 2022, from <https://xunit.net/>
- [16] NPM. (n.d.). *About npm*. npm Docs. Retrieved May 16, 2022, from <https://docs.npmjs.com/about-npm>
- [17] Sommerville, I. (2011). Functional and non-functional requirements. In *Software Engineering* (9th ed., pp. 84-91). Pearson. 978-0-13-705346-9

- [18] Angular. (n.d.). *What is Angular? - link*. Angular. Retrieved May 16, 2022, from <https://angular.io/guide/what-is-angular>
- [19] getBootstrap. (n.d.). *Bootstrap*. Bootstrap · The most popular HTML, CSS, and JS library in the world. Retrieved May 16, 2022, from <https://getbootstrap.com/>
- [20] Smith, S. (2022, March 25). *Overview of ASP.NET Core MVC*. Microsoft Docs. Retrieved May 16, 2022, from <https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0>
- [21] Wikimedia. (2021, juli 4). *Wikimedia*. Mvc-Process Wikimedia. <https://commons.wikimedia.org/w/index.php?curid=10298177>
- [22] JSON. (n.d.). JSON.org. Retrieved May 16, 2022, from <https://www.json.org/json-en.html>
- [23] JWT. (n.d.). *JSON Web Token Introduction - jwt.io*. JWT.io. Retrieved May 16, 2022, from <https://jwt.io/introduction>
- [24] Python Cryptographic Authority. (n.d.). *Hash-based message authentication codes (HMAC) — Cryptography 38.0.0.dev1 documentation*. Hash-based message authentication codes (HMAC) — Cryptography 38.0.0.dev1 documentation. Retrieved May 16, 2022, from <https://cryptography.io/en/latest/hazmat/primitives/mac/hmac/>
- [25] *Regular expressions - JavaScript | MDN*. (2022, May 6). MDN Web Docs. Retrieved May 16, 2022, from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions
- [26] Sommerville, I. (2011). Software validation. In *Software Engineering* (9th ed., pp. 41-43). Pearson. 978-0-13-705346-9

[27] Edureka. (2020, November 25). *Software Testing Models | 6 Types of Software Testing Models*. Edureka. Retrieved May 16, 2022, from <https://www.edureka.co/blog/software-testing-models/#vmodel>

[28] NUnit.org. (n.d.). *What is NUnit?* NUnit.org. Retrieved May 18, 2022, from <https://nunit.org/>

International Software Testing Qualifications Board. (2019, April 28). *ISTQB Grunnivå pensum*. Norwegian Testing Board. Retrieved May 16, 2022, from <http://www.istqb-norge.no/wp/wp-content/uploads/2019/04/CTFL-2018-Pensum-norsk-v1.0.pdf>

Schaefer, H. (n.d.). *Systematisk Testing av Software*. Norwegian Testing Board. Retrieved May 16, 2022, from <http://www.istqb-norge.no/wp/wp-content/uploads/2014/02/Systematisk-Testing-av-Software-intro.pdf>

Sommerville, I. (2011). Context models. In *Software Engineering* (9th ed., pp. 121-124). Pearson. 978-0-13-705346-9