Bachelor's thesis

Rune Schei Bergheim
Martin Nauf Staer
Sverre Ose Dybdahl
Snorre Kvalvåg Dahlen

Børs Buddi

Et hjelpeverktøy for aksjehandlere

May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Bachelor's thesis

2021

NTNU
Norwegian University of
Science and Technology

Rune Schei Bergheim
Martin Nauf Staer
Sverre Ose Dybdahl
Snorre Kvalvåg Dahlen

Børs Buddi

Et hjelpeverktøy for aksjehandlere

Bachelor's thesis
May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU

Kunnskap for en bedre verden

Bacheloroppgave

IE303612 Bacheloroppgave Data

Børs Buddi - et hjelpeverktøy for aksjehandel

Snorre Kvalvåg Dahlen, Sverre Ose Dybdahl, Martin Nauf Staer, Rune Schei Bergheim

Totalt antall sider inkludert forsiden: 62

Ålesund, 20.05.2021

NTNU

Kunnskap for en bedre verden

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.	☒
2.	Jeg/vi erklærer videre at denne besvarelsen: • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§14 og 15.	☒
4.	Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver	☒
5.	Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens studieforskrift §31	☒
6.	Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider	☒

Figurer

Innholdsfortegnelse

SAMMENDRAG

Dette prosjektet går ut på å lage et IT-system som gir beslutningsstøtte ved kjøp og salg av aksjer på børsen, basert på innsamlede artikler og analyse av tidligere transaksjoner. Ved hjelp av Scrum og designprinsipper for brukervennlighet og kodestruktur har systemet, Børs Buddi blitt konstruert. Systemet er web basert, og inneholder funksjonalitet som analyserer transaksjoner og samler inn artikler fra en nettavis. I rapporten er det redegjort for utviklingsprosessen, teknologier og resultater fra prosjektarbeidet. Resultatene er drøftet, og forslag til forbedringer og videreutvikling er presentert. Gjennom arbeidet med systemet er det lagt til rette for mulig videreutvikling av artikkel-analyser.

TERMINOLOGI

Begreper

Open source - Software der koden er tilgjengelig for bruk eller modifikasjon

Portefølje - Samling av verdipapirer

JSON - Fil format for elektronisk datautveksling som er leselig for mennesker ved bruk av tekst for lagring og sending av objekter i en liste.

Kurtasje - Avgift som går til aksjemegler ved kjøp og salg av aksjer.

API - Hjelpeverktøy som har beskrivende funksjonalitet til en database, operativsystem eller lignende.

Forkortelser

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

HTTP - Hypertext Transfer Protocol

API - Application programming interface

XML - Extensible Markup Language

XPath - XML Path Language

RSS - Really Simple Syndication

NLP - Nevro Lingvistisk Programmering

IDE - integrated development environment.

CSV - Comma-separated values

TSV - Tab-separated values

JSON - JavaScript Object Notation

SRP - Single Responsibility Principle

OCP - Open-closed Principle

LSP - Liskov Substitution Principle

ISP - Interface Segregation Principle

SDK - Software development kit

TCP - Transmission Control Protocol

1 INNLEDNING

1.1 Bakgrunn

Vi hører ofte i mediene om store pengesummer som tjenes eller tapes på aksjehandel både hos enkeltpersoner og hos store hedgefonds. Håpet om en rask gevinst gjør aksjemarkedet svært attraktivt for håpefulle mennesker som ser muligheten til å tjene penger ved bare noen få tastetrykk på pc'en.

Under korona-pandemien har interessen for aksjemarkedet økt kraftig på grunn av lave bank-renter og ønske om bedre avkastning på ledig kapital. For eksempel har aksjemegler-plattformen Nordnet sett en økning av brukere på 53 500 bare i første kvartal av 2021 (Hartwig, 2021). Dette betyr at flere investerer og konkurransen i markedet øker. I en travel hverdag har ikke alle småsparere på aksjemarkedet tid til å fordype seg i nyhetssaker og analyser som kan gi dem en fordel og økt avkastning på børsen.

Utfordringen for de fleste med lite erfaring i markedet er at store aktører har tilgang på avanserte analyseverktøy som gir de en mye bedre innsikt i markedet enn de små aktørene (Thomas, 2019). og derfor kan gjøre bedre beslutninger. Det blir ugunstig for de små aktørene å kjøpe dyre verktøy, siden profittmarginen da blir minimal. Det er derfor et behov i markedet for et verktøy tilpasset små-investorer, hvor man kan spare tid, der kostnaden er lav og hvor man kan få god beslutnings-støtte når man vurderer å kjøpe/selge enkelt-aksjer på børsen.

1.2 Problemstilling

Lage et webbasert system som skal redusere tid for innhenting av artikler og enkel analyse av spesifikke handler som skal hjelpe brukeren ta mer informerte beslutninger for kjøp og salg av aksjer i framtiden.

1.3 Målsetting

Målene for prosjektet er som følger:

1. Utvikle et besluttningsstøtte-system for aksjehandlere, der bruker kan ta mer informerte valg ved fremtidige kjøp og salg på børs basert på analyse av brukerens tidligere transaksjoner.
2. Systemet skal hente inn artikler fra nettaviser relevant til aksjemarked.

3. Legge til rette for videreutvikling av systemet slik at artikler får en karakter basert på en digital analyse, som viser hvor positive eller negative artikkelen er til selskapet de handler om.

1.4 Avgrensning

Nordnet er oppdragsgivers hovedplattform for aksjehandel. Forfatterne valgte av den grunn å kun støtte loggfiler fra Nordnet da det er en tidkrevende prosess å produsere et skript som leser og henter ut data, samtidig at det er vanskelig å teste de grunnet manglende testdata fra andre kilder enn Nordnet.

Det bør ikke koste mye å drive denne applikasjonen ettersom markedet vil bestå av småsparere som ikke kan betale for mye. Derfor begrenser det muligheten til å ta i bruk APIer eller andre verktøy.

I utgangspunktet var det et ønske fra oppdragsgiver at systemet skulle kunne støtte utenlandske aksjemarkeder og språk. Dette ble avgrenset til kun norsk, da kostnadsproblemer kunne oppstått på lik linje som andre nyhetsportaler. Det ville også ført til en betydelig større arbeidsmengde, og loggboksystemet ble etter samtaler med arbeidsgiver prioritert ovenfor ekstra markeder og språk.

Å lage et analyseverktøy som garanterer et riktig resultat hver gang er en oppgave som fort kan bli veldig kompleks og tidkrevende. Derfor ble avgjørelsen tatt å forenkle dette til å heller være et verktøy som kan lede bruker til interessante artikler, men ikke ta en avgjørelse for bruker. Et alternativ ville også vært å bruke et eksternt system for sentimentanalysen, men siden systemet helst skal være kostnadsfritt måtte det avgrenses slik.

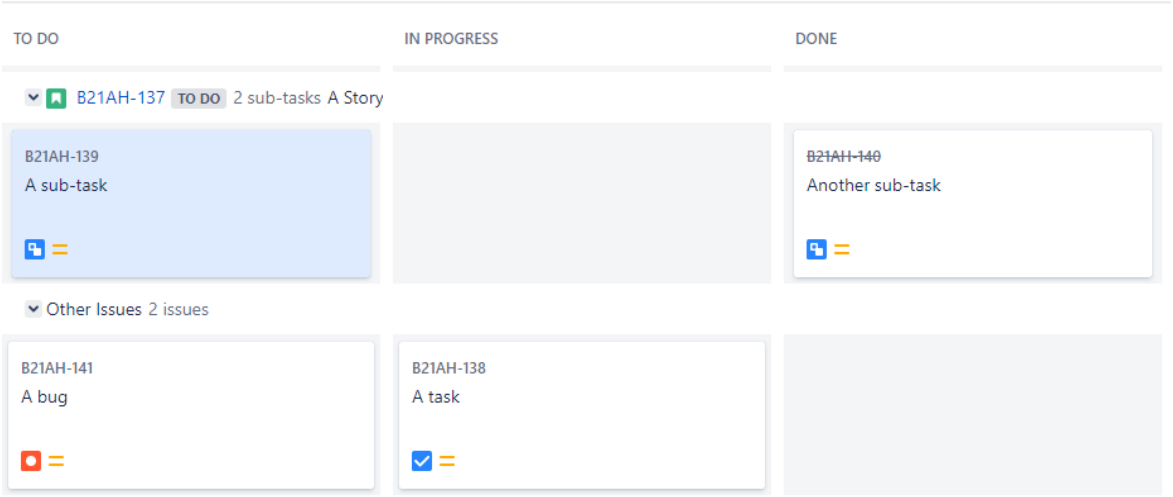
2 TEORETISK GRUNNLAG

Dette kapittelet inneholder teorien som er utnyttet for å utvikle løsningen på problemstillingen, og skal gi leseren nok kunnskap til å forstå det som kommer senere i rapporten.

2.1 Scrum

I programvareutvikling er det normalt å bruke smidige utviklingsmetoder, en av disse er Scrum. Scrum er et rammeverk for prosjektledelse for å optimalisere produktutvikling ved å legge vekt på samarbeid, ansvar og iterative prosesser mot et definert endemål. Dette rammeverket tillater høy grad av tilpasning av prosjektet i løpet av prosjektets levetid, tydelige mål og høy grad av samarbeid i teamet samt med kunde (TechTarget Contributor, 2017)

2.1.1 Scrum-board



Figur 1- Scrum board

Eksempel på et Scrum-board med stories, tasks, subtasks og en bug.

Scrum-board er en visuell representasjon av arbeidet som skal gjøres hver sprint. Figur 1. Et Scrum-board inneholder minimum stories (oppgaver), som er plassert under en av de følgende tilstandene" to do", "under arbeid" og "ferdige". Hver story eller underoppgave til en story blir tildelt en utvikler, slik at hver utvikler lett kan se hvilke oppgaver de har igjen og gjøre og hvilke oppgaver som er ferdige (TechTarget Contributor, 2017).

2.1.2 Sprint

En sprint er tidsrommet hvor et bestemt antall oppgaver skal bli utført. Hver sprint består av flere møter og varer vanligvis i 1 til 4 uker (TechTarget Contributor, 2017):

- Sprint planleggingsmøte hvor alle i teamet er med på å sette mål for å nå minst en forbedret versjon av produktet.
- Daglige møter hvor man har et kort møte hver dag der man diskuterer hva man har gjort i løpet av forrige døgn og hva som skal være gjort det neste i løpet av døgnet.
- Sprint review hvor man viser fram hva man har produsert i løpet av sprinten.
- Sprint retrospektive hvor hele teamet reflekterer på sprinten og vurderer hva som har gått bra og hva som kan forbedres.

2.1.3 Scrum-roller

Scrum består av tre roller; produkteier, Scrum master og utviklingsteamet (TechTarget Contributor, 2017).

1. Produkteieren jobber som en mellommann mellom kunde og utviklerteam, og har ansvaret for at forventningene til sluttproduktet er tydelig for resten av teamet, samt at det eksisterer en visjon for produktet.
2. Scrum masteren er et medlem av utviklingsteamet og har ansvaret for å tilse at de beste praksisene blir brukt, at prosjektet beveger seg framover, primærkontakt mot kunde og å hjelpe til med avklaringer og prioriteringer av oppgaver.
3. Utviklingsteamet er profesjonelle utviklere som jobber sammen som en gruppe for å lage og teste hvert ledd i utviklingsprosessen.

2.2 Git

Git er et open-source kommandolinjeverktøy som brukes til versjonskontroll. Det eksisterer flere programmer som implementerer Git, eksempelvis GitHub Desktop og Sourcetree, mens JetBrains har en egen utvidelse implementert. Dette kapittelet inneholder teori om hvilke funksjoner dette verktøyet har og om hvordan GitHub Desktop fungerer.

2.2.1 GitHub

GitHub er en plattform som implementerer Git i et web-basert grafisk brukergrensesnitt og tillater flere parter å samarbeide på et felles prosjekt. For enkel tilgang til GitHub så eksisterer det et eget program som kan lastes ned på en datamaskin, GitHub Desktop. Dette programmet holder kontroll over mappen med alle filene til systemet som utvikles og man

kan branche, utføre pull request og pushe endringer ut på den delte mappen som ligger på GitHubs nettsider.

2.2.2 Branch

En branch(gren) i GitHub er en selvstendig linje med utvikling som er basert på hovedgrenen. En gren splittes som regel ut ifra hovedgrenen når ny funksjonalitet skal kodes, dette tillater store endringer og testing av ny funksjonalitet uten å endre på den forrige stabile versjonen.

2.2.3 Pull Request

Pull-requests brukes for å gå igjennom ny kode før den integreres med resten av koden. Man kan se på dette som et valgfritt godkjennings-steg der andre utviklere får muligheten til å se endringene som er gjort og kan be om forbedringer eller nye endringer før den integreres med resten av koden. Dette kan også brukes til å oppdage feil og for å opprettholde den eksisterende kodens kvalitet og struktur. (BEKK Open Radar, 2018)

2.3 Design prinsipper

De følgende prinsippene er anbefalt å følge for å opprettholde gode funksjoner, kodestruktur og brukervennlighet.

2.3.1 Don Normans 6 brukbarhets prinsipp

Don Normans 6 brukbarhets prinsipp er prinsipper som gir en pekepinn på hvordan man bør tenke når man designer noe, prinsippene er:

1. Visibility: En funksjon eller et valg må være synlig for bruker, dette gir en høyere sannsynlighet for at brukeren kommer til å bruke den, da brukeren skjønner at den er der.
2. Feedback: Når en funksjon blir aktivert er det viktig med tilbakemelding slik at brukeren er klar over at funksjonen har blitt aktivert, dette kan gjøres på flere måter som, synlig tilbakemelding eller lyd.
3. Constraints: Ikke presentere unødvendige valg for en bruker, begrense mulighetene til bare det mest nødvendige.
4. Mapping: Plassering av en kontrollknapp bør være logisk opp mot effekten denne knappen har.
5. Consistency: Like symboler for like funksjoner og motsatt, ikke ulike symboler for lik funksjonalitet. Grønn for ja, positivt eller kjør og rødt for nei, negativt eller stopp.
6. Affordance: Utforming skal gi et hint om funksjonaliteten.

(Enginess, 2014)

2.3.2 Objektorientert design

SOLID er et akronym for fem prinsipper som blir brukt ved objektorientert programmering, de fem prinsippene er:

1. **S**ingle Responsibility Principle (SRP) er prinsippet at en klasse skal ha et ansvar, det vil si at et skript som utfører flere oppgaver ikke bør ha disse oppgavene fordelt over flere funksjoner.
2. **O**pen-closed Principle (OCP) er prinsippet for at et objekt skal være åpen for utvidelse, men ikke for modifisering. Dette vil si at det bør være mulig å utvide klassen uten å modifisere klassen i seg selv, med unntak for bugfiksing.
3. **L**iskov Substitution Principle (LSP), dette prinsippet omhandler at en underklasse skal kunne erstatte en superklasse. Det vil si at en underklasse ikke kan fjerne, men bare legge til ekstra funksjonalitet.
4. **I**nterface Segregation Principle (ISP) er prinsippet med å dele opp større grensesnitt ned til mindre grensesnitt, slik at man kan implementere de grensesnittene som er viktig for seg selv og ikke de grensesnittene som er viktige for andre.
5. **D**ependency Inversion Principle, dette prinsippet omhandler det å holde lav coupling mellom moduler i programvare slik at ikke høy-nivå moduler er avhengige av lav-nivå moduler.

(Baeldung, 2020)

I tillegg til disse prinsippene er det et spesielt prinsipp som er viktig å følge, “Don’t Repeat Yourself” (DRY). Dette er prinsippet for å unngå duplikat kode, slik at man ikke har flere identiske funksjoner eller konstanter i systemet. Ved å unngå duplikat kode i et system er det enklere å vedlikeholde systemet ettersom man kan være sikker på at denne funksjonen eller konstanten er endret alle plassene den er brukt ved å endre den en gang

(Paul, 2020)

2.4 Frontend

Fronten er den delen av programmet som ligger nærmest brukeren. Den bygges opp av HTML som er siden brukeren ser, CSS som angir stylingen på siden og JavaScript som gir siden interaktiv funksjonalitet.

2.4.1 Programmeringsspråk

2.4.1.1 HTML

HyperText Markup Language (HTML) er et standardisert programmeringsspråk som beskriver strukturen til en nettside for en nettleser. Strukturen på denne koden består av flere elementer hvor hvert element er avgrenset med en spesifikk tag (nøkkelord). Hver tag har en spesifikk oppgave og brukes avhengig av hva innholdet til det elementet skal være (Christensson, 2015).

2.4.1.2 CSS

Cascading Style Sheet (CSS) er programmeringsspråket som forteller nettleseren hvordan designet på nettsiden skal se ut. Dette gjøres ved å kode designet for hvert HTML-element. Designet inneholder alt fra posisjon av elementet, teksttype, farge og bakgrunnsfarger.

2.4.1.3 JavaScript

Javascript er et programmeringsspråk som tillater nettsider å reagere på hendelser uten å skifte sider. Dette gjøres ved å lage funksjoner som "lytter" etter hendelser som museklikk eller respons fra server side og deretter kjører funksjonen sin.

2.5 Backend

Backend er delen av et system som brukere ikke har noen direkte interaksjon med. Dette er i mange tilfeller serverløsninger, databaser, og andre systemer som ikke legges merke til av bruker.

2.5.1 Programmeringsspråk

2.5.1.1 Java

Java er et klasse- og objektorientert programmeringsspråk som er designet med så få implementerte avhengigheter som mulig. Dette språket er designet slik at alle plattformer som støtter Java skal kunne kjøre den kompilerte koden man har skrevet (Christensson, 2012)

2.5.1.2 Python

Python er et høy level programmeringsspråk som er designet for lesbarhet og er avhengig av innrykk og nye linjer for avsluttning av funksjoner. Dette er et språk med store mengder biblioteker innebygget, slik at man kan finne de fleste funksjonene man trenger uten å være avhengig av å laste ned eksterne biblioteker (Python Software Foundation, 2021).

2.5.1.3 Dockerfile

Dockerfile er programmeringsspråket som forteller Docker, 2.5.5 Server , hva den skal gjøre når det blir startet. Dette skrives inn i en Docker fil og beskriver alt fra hvilke filer som skal være med, hvor de er plassert, hvor de skal og hvilke biblioteker som skal være med.

2.5.2 Rest

REST er en lettvektig og universell måte å opprette kommunikasjon mellom klient og server. Hovedelementene i REST er ressurser, spørreord (HTTP-metoder), forespørsler og respons. (GURU99, u.d.)

2.5.2.1 Ressurser

Ressurser er informasjon som ligger lagret på en server, som en ønsker å hente ut eller modifisere ved hjelp av REST arkitekturen. Eksempler på ressurser kan være tekst og bilder.

2.5.2.2 HTTP-metoder

Det finnes fire HTTP-metoder som definerer hvordan ressurser skal behandles.

1. GET: Klienten ønsker å hente ut ressurser, men ikke å gjøre noen endringer.
2. POST: Klienten ønsker som regel å opprette ny informasjon som skal lagres i en database.
3. PUT: Klienten ønsker å endre på allerede eksisterende ressurser.
4. DELETE: Klienten ønsker å slette ressurser fra databasen.

2.5.2.3 HTTP-Forespørsel

En forespørsel brukes til å fortelle serveren hva som skal gjøres. Et kall blir definert med et HTTP-metode, adresse, parameter, header og body.

URI (Uniform Resource Identifier) Blir brukt av REST arkitekturen til å finne ressursen som blir spurt etter. På klientsiden sendes kall basert på URI, da vil en metode som er tildelt følgende URI bli utført på serversiden. (Restfulapi, u.d.)

To **parametere** brukes ofte for å få tilgang på mer spesifikk informasjon.

- Query parameter brukes gjerne til å filtrere responsen en får etter en spesifisert query, den er spesifisert ved å sette et spørsmålstegn etter URI'en. (Restfulapi, u.d.) Et eksempel på dette er: https://www.google.com/search?q=google

- Path parameter brukes til å definere en spesifikk ressurs som skal hentes ut. Den er spesifisert ved å sette en id eller en annen form for identifisering i krøllparentes etter URI'en. (Rapidapi, u.d.) Her er et eksempel: example.com/api/users/{userid}

Body er selve innholdet, som kan være en ren tekst, formatert tekst (et eksempel er JSON), eller form-data m.m. Body er som oftest sendt sammen med POST eller PUT metoder, der det er behov for å sende med data som skal endres eller opprettes.

Header er tilleggsinformasjon som kan være en beskrivelse av innhold (eksempelvis lengde av innhold og type)

2.5.2.4 Respons

Response body viser selve svaret fra server, eksempelvis i et JSON format, eller det kan komme i form av HTML kode, det får man ofte dersom man får en feilkode, for å vise feilkoden i nettleseren. I noen tilfeller brukes ikke response body, det kan være nok med kun respons kode.

Statuskoder brukes til å fortelle nettleser og applikasjonen hva som skjedde med forespørselen som har blitt sendt. De brukes ofte på klientsiden til å vise feilmeldinger, eller for å la applikasjonen forstå at den kan jobbe videre med det den holdt på med.

Eksempler på statuskoder en ofte ser:
200 = ok
401 = uautorisert
404 = finner ikke informasjonen
500 = feil oppstod på server

2.5.3 Databaseteori

Databaser er en strukturert måte å lagre og vedlikeholde data på. Det brukes en DBMS (Database Management System) for å manipulere innholdet, og for å styre sikkerhetsfunksjonalitet. (Guru99, u.d.)

2.5.3.1 Query

En query kan defineres som en instruks en sender til databasen for å hente ut eller modifisere spesifikk data. Da defineres først en operasjon en ønsker å utføre (eks SELECT, INSERT, DELETE) fulgt opp med hvilken tabell en ønsker å adressere data i, og til slutt en filtrering for å spesifisere hvilke data i tabellen som skal manipuleres eller hente ut. (Dette er et enkelt

eksempel på en query, det finnes mange tilfeller der dette ikke er tilstrekkelig). (Williams, 2016)

2.5.3.2 Skjema

Et skjema (schema) er et «kart» som viser databasens struktur. I et slikt skjema kan en se hvilke tabeller som er med, og forholdet de har med hverandre i form av primary og foreign keys. (Williams, 2016)

2.5.3.3 Nøkler

En bruker som oftest to forskjellige typer nøkler i tabellrader. Disse nøklene har som funksjon å bli brukt til å identifisere unike trekk ved radene så en alltid har en måte å skille de.

Primary key er den nøkkelen som unikt identifiserer en tabell rad. For en bruker kan det for eksempel være et brukernavn, eller en generert id som blir gitt til den av systemet.

Foreign key er en nøkkel som er en primary key i en annen tabell, ved å opprette den som primary key i en annen tabell opprettes det en relasjon mellom de tabellene. Og en kan dermed bruke foreign key til å hente inn informasjonen som tilhører den i den andre tabellen.

Et eksempel for en nettbutikk:

En bruker har en tabell med id (primary key), navn og mail og annen relevant informasjon. Videre er det en tabell for annonsene, som har en egen id (primary key), selgerens id (foreign key), og annen relevant informasjon.

2.5.4 Webserver

Maven er et verktøy som blir brukt til å importere biblioteker til prosjektet på en enkel og oversiktlig måte. Tradisjonelt vis så vil en laste ned biblioteker fra forskjellige nettsteder og manuelt legge den inn i prosjektet. Men Maven tar seg av dette, ved hjelp av at en definerer hvor og hva den skal hente. Dette fører til et ryddig og oversiktlig system, samtidig som at det forenkler oppgaven å dele prosjektet med andre. (Apache Maven, 2021)

Eclipse Microprofile har som formål å være et hjelpeverktøy for å implementere «microservices» i et prosjekt. (Monteiro, 2018). Microservices er en måte å sette opp prosjektet i flere enkeltstående moduler. Det vil si at service klasser (de klassene som behandler http forespørsler) opererer uavhengig av hverandre. (Turizo, 2019)

JAX-RS er en api som gjør det enklere for utviklere å utvikle RESTful web services. Den gjør det ved å la utvikler lage «Plain Old Java Objects» eller «POJO» for ressurser via requests. Det fungerer slik at når det mottas en request så gjøres det om til en POJO som lagres i databasen. Videre når en skal gi en respons så transformeres den fra et POJO til en http response. (Gilstrap, 2009).

Dette tilbyr JAX-RS via en rekke annoteringer som brukes for å spesifisere innhold og handling i et forespørsel eller respons.

Noen av disse annoteringene er:

- @GET, @POST, @PUT, @DELETE – som http request type. Altså hvilken type handling som skal brukes for ressursen.
- @PRODUCES – spesifiserer hvilken mediatype responsen skal ha, en ofte brukt response type er JSON
- @CONSUMES – spesifiserer hvilken mediatype den aksepterer at requesten har. Det er ofte form data eller JSON

I tillegg har den annoteringer som spesifiserer parametere, i form av formparameter, query parameter eller annet. Dette er som regel brukt til å filtrere ut relevant informasjon når det søkes gjennom databasen. (Wikipedia, 2021).

Jersey er en utvidelse for JAX-RS med mål om å videre forenkle og optimalisere RESTful Web Services. (Eclipse Foundation, u.d.)

Project Lombok er et bibliotek som gjør det mulig å fjerne en del kode til fordel for annoteringer i stedet. For eksempel kan en fjerne tom constructor (som er nødvendig å ha i Java EE) og heller skrive @NoArgsConstructor for å spesifisere at det skal være med. Dette kan gjøres med getters, setters og andre elementer. Det skaper bedre oversikt over prosjektet og gjør det enklere å skrive. (Baeldung, 2020)

JSON Web Token brukes som et autorisasjons verktøy. Når en logger på en nettside så genereres en token som blir sendt tilbake og ofte lagret av klienten. Videre når det oppstår situasjoner der autorisering vil være nødvendig, da for eksempel ved sensitiv informasjon, vil JWT sendes med forespørselen til server som en enklere måte å autorisere brukeren enn å måtte skrive inn brukernavn og passord manuelt. (JWT, u.d.)

2.5.5 Server

Ubuntu Server er et server operativsystem. Det er et av de mest vanlige operativsystemene for serverløsninger, og det fungerer på nesten alle maskiner en kommer over. Dette gjør Ubuntu Server til et kostnadseffektivt og lett tilgjengelig server-operativsystem. (Wallen, 2020).

Docker opererer ved å legge applikasjoner i "kontainere" som er spesialbygde plattformer applikasjonene kjøres på. Dette bidrar til enkelhet for utviklere da en vet at slike kontainere vil virke på andre operativsystem som også kjører Docker. Samtidig er kontainerene relativt lettvektige sammenlignet med alternativet som er virtuelle maskiner med et helt operativsystem for å kjøre hver applikasjon eller service. Applikasjoner konfigureres enkelt i en Dockerfil, og i en Docker compose fil kan en spesifisere hvordan flere konteinere skal fungere sammen.

Cron er et system som brukes for å opprette tidsbaserte «jobber». Disse jobbene er i form av shell kommandoer på Unix baserte operativsystem. For å opprette en jobb lager man en crontab (en tekstfil med en spesiell syntax for cron) der man spesifiserer hvilken kommando som skal utføres og når.

Payara er en web server løsning for Java EE applikasjoner. Det er en forbedret versjon av glassfish, men den er hyppigere oppdatert og har ytterligere funksjonalitet. Glassfish er referanseimplementasjonen for Java EE systemer, som i korte trekk betyr at det er standard implementasjonen og at skal støtte all funksjonalitet i Java EE.

NGINX er et verktøy med flere funksjoner, kan brukes som en web server eller «reverse proxy» (NGINX, u.d.) Kort sagt betyr reverse proxy det at den opererer som et mellomledd for web klienten og web serveren (CLOUDNWEB, u.d.). Dette kan bidra til et bedre system ved å balansere lasten mellom forskjellige servere, beskyttelse mot angrep da en ikke trenger å synliggjøre IP-adressen til serveren, caching i NGINX serveren som minsker lasten på web serveren, og kryptering av innkommende og utgående kall og responser som også kan minske lasten på web serveren (Cloudflare, u.d.). Disse fordelene merkes best på et større system.

Postman gjør kall til serverens API. En konstruerer API kall som simulerer de som JavaScript ville sendt ved å legge til egendefinerte parameter. Det finnes en rekke funksjoner som Postman til et verdifullt verktøy. Det leverer en ryddig måte å se respons fra server, programmerbare variabler slik som brukernavn, passord og tokens. Og den har muligheten automatisere rekker av API kall som sparer mye tid.

PuTTY er en open-source terminal emulator. Det tilbyr en trygg og enkel måte å koble seg opp mot et UNIX basert operativsystem fra en Windows klient. (Perrin, 2008). Noe som PuTTY tillater som f.eks Windows PowerShell ikke gjør er å lagre tilkobling konfigurasjoner, som kan være en stor fordel når en bruker systemet ofte. (Cawley, 2018)

FileZilla er en applikasjon som gjør det enkelt og smidig å overføre filer til og fra en server. Ved hjelp av IP adresse, navn og passord for brukeren, og portnummer vil FileZilla opprette en SFTP kobling automatisk. Deretter kan brukeren dra og slippe filer fra og til server som ønsket.

H2 Database er en lettvektig Java-basert databaseløsning. Den kan bli konfigurert som en "in memory database" som altså betyr at den ikke lagrer dataene på disk (tutorialspoint, u.d.). Dette gjør den utrolig rask, men på bekostning av at dataen kan forsvinne dersom systemet skrur seg av. H2 er derfor best utnyttet i test og produksjonsfaser. (tutorialspoint, u.d.)

2.6 Datainnsamling

Datainnsamling (data collection) er prosessen med å samle inn data på en ordnet systematisk måte, slik at den kan analyseres og brukes ved en senere anledning. Dette utføres på mange forskjellige måter avhengig av hvilken data man samler inn og hva den skal brukes til.

2.6.1 Søkerobot

En søkerobot (web crawler) er en robot som systematisk søker gjennom the World Wide Web og utfører oppgaver de er programmert til. Disse robotene blir typisk brukt av søkemotorer til å søke rundt på nett for å finne, kopiere og indeksere nettsider. Sidene som blir kopiert blir så prosessert av søkemotorene for å oppdatere innholdet eller indeksene til sine og for at brukere skal kunne søke mer effektivt etter relevante sider.

Søkeroboter bruker ressursene til nettsidene de besøker og det finnes derfor mekanismer for nettsider som ikke ønsker at denne trafikken. Et eksempel på en slik mekanisme er å inkludere en robots.txt fil som ber roboten om å ikke indeksere siden eller bare deler av siden (Jackson, 2017).

2.6.2 Nettskraping

Nettskraping (web scraping) er en robot basert på en søkerobot som systematisk surfer the World Wide Web og kopierer data som blir lagret til senere bruk eller analyse. Nettskraping fungerer typisk på en av to måter, via Hypertext Transfer Protocol (HTTP) eller via en nettleser, og består typisk av to prosesser henting og ekstrahering. Når man henter en nettside

så blir denne lastet ned for senere prosessering. Når dette er gjort ekstraherer man dataen man er ute etter på denne siden.

Nettskraping blir brukt i flere sammenhenger som forskning, data mining, prissammenligning og kontakt skraping. Ikke all bruk av nettskraping er lovlig og det er derfor viktig å være sikker på hva det er man samler inn da man må følge nasjonale og internasjonale lover ved for eksempel kontakt skraping, som er å samle inn e-poster for så å samle inn personopplysninger.

Det er og flere teknikker for nettskraping som HTTP programmering, DOM analysering og HTML analysering. HTML analysering er teknikken for å skrape nettsider som blir generert dynamisk basert på en underliggende datastruktur som en database. Data i like kategorier blir da kodet likt basert på en mal. Nettskraperen vet da fra programmering hvilken data den skal hente ut og hvor den finner dataen i HTML koden (Wikipedia, 2021).

2.6.3 Sentiment analyse

Sentimentanalyse (opinion Mining) er bruken av naturlig språk prosessering, tekst analyse, beregning lingvistikk og biometri for å systematisk identifisere og ekstrahere subjektive informasjon. Dette brukes i flere sammenhenger som i kundeanmeldelser, undersøkelser og på sosiale medier for å finne ut hva den underliggende meningen med tilbakemeldingen betyr. Det er og forskjellige typer sentimentanalyse som å finne polariteten i teksten, objektiv/subjektiv identifisering og trekk/aspekt-basert.

Å finne polariteten på en tekst kan gjøres ved en enkel sammenligning av negative og positive ord, men denne metoden gir nødvendigvis ikke det riktige resultatet da det kan være flere positive ord i en negativ tekst og flere negative ord i en positiv tekst. En annen metode er at hvert positive og negative ord har en vekt som blir brukt til å beregne den totale polariteten i teksten.

2.6.4 RSS

Really Simple Syndication (RSS) er et eldre system for abonnering på et nettsteds RSS-feed, RSS-feeden inneholder nyheter, artikler og podcasts fra nettstedet som man kan hente ut uten å måtte besøke nettstedet. Dette systemet sender ut artikler fortløpende i et XML-format, med lenket til innholdet, tittel, ingress og potensielt bilder og lyd. Denne teknologien var meget populær på starten av 2000-tallet, men populariteten har siden avtatt og ikke alle nettlesere har støtte for RSS lengre. For å utnytte seg av en RSS-feed må man ha en RSS-leser, en RSS-

leser er et simpelt script som leser igjennom en RSS-feed og henter ut den informasjonen som er ønsket (Rossen & Øverby, 2020).

2.6.5 XPath

XML Path Language (XPath) er et spørrespråk (query language) for å velge noder i et XML-dokument, XPath kan også brukes til å beregne verdier fra innholdet i et XML-dokument. XPath språket er basert på en tre representasjon av XML-dokument og gir muligheten til å navigere et XML-dokument og velge noder fra spesifikke kriterier. XPath 1.0 ble anbefalt av World Wide Web Consortium (W3C) i 1999 og har siden den gang blitt oppdatert til versjon 3.1 som ble anbefalingen fra 2017 (Robie, et al., 2021).

2.7 Børs

Handelsplass for kjøp og salg av aksjer der handler administreres som oftest av en aksjemegler. Her kan det også bli utdelt utbytte for aksjer basert på resultatet et selskap gjør. Børsen holder oversikten over hvert enkelt selskaps aksjeverdi basert på handler som blir gjort. Et offentlig selskap vil være notert på børsen i dens respektive tilhørighet. Det er vanlig for norske firmaer å være notert på Oslo børs og vil dermed bli mest omtalt i norske medier. (Wikipedia, 2021)

2.7.1 Aksjemarked

Aksjemarkedet er der handel mellom kjøper og selger finner sted og representerer eierskapet av forskjellige selskaper, dette innebærer både private og offentlige selskaper.

2.7.2 Nordnet

Aksjemegler som baserer seg i Skandinavia og som tilbyr kjøp av fond, aksjer og spareprodukter. Tjenester som gir tilgang til børsen i mange forskjellige land, gjør det mulig for enkeltpersoner å investere i selskaper over hele verden. Hos Nordnet har man oversikt over porteføljen sin og hvordan investeringer minker eller vokser.

2.8 Eksisterende løsninger

I dette underkapittelet skal vi gå igjennom hvilke eksisterende løsninger som har blitt vurdert å bruke i dette prosjektet.

2.8.1 CSV til JSON

For å importere CSV filer og konvertere dem til JSON objekt finnes det en rekke løsninger og tutorials.

GeeksforGeeks løsning leser først filen med en fs npm pakke før den gjør om filens innhold

til en streng som splittes inn i en liste for hver rekke i tabellen. Deretter genereres en liste med alle headers i en liste. For de resterende rekkene legges strengen deres inn i hver sitt objekt, og strengene splittes opp i flere strenger slik at hver verdi blir en streng. Så legges headers inn i alle objektene til sin tilhørende verdi. Til slutt konverteres listen med objektene til JSON (Karankc27, 2021).

En annen måte å gjøre dette på er fra gitconnected hvor de bruker Javascript biblioteket XLSX. Først brukes en file-reader til å lese regnearket som en binær streng, så brukes bibliotekets innebygde funksjon for å konvertere den binære strengen til et JSON objekt (babu, 2019).

2.8.2 Loggbok

For å ta imot en JSON liste og printe den ut som en tabell på nettsiden, finnes det flere forskjellige løsninger. Den mest relevante løsninger er å lage en dynamisk tabell i Javascript, fordelen med en dynamisk liste vil være at ettersom listens størrelse vil variere må tabellens størrelse også kunne variere.

2.8.3 Nettskraper

Det eksisterer flere forskjellige løsninger og rammeverk for nettskraping på nettet. De følgende løsningene er ferdige programvarer som man konfigurerer ved hjelp av et grafisk brukergrensesnitt, mens de følgende rammeverkene er tilpasset Java eller Python og krever noe undersøkelse i hvordan man programmerer de.

Octoparse er en ferdig løsning som har en gratisversjon, men prisen er $209 i måneden for den dyreste løsningen. Gratisversjonen har en begrensning på 10 000 poster per eksport, 10 nettskrapere, ingen automatisert tidsstyring og lite brukerstøtte, mens den dyreste løsningen er begrenset på 250 crawlere og har mange flere funksjoner bygget inn. Resultatene fra en skraping av denne løsningen kan lastes ned som CSV, Excel, via API eller lagres direkte til en database (octoparse, 2020).

Parsehub er en nettskraper som og har en gratisversjon, mens den dyreste løsningen deres koster $499. Gratisversjonen er begrenset med 200 sider per kjøring på 40 minutter og prosjektene man lager i gratisversjonen blir offentlig. Den dyreste løsningen tilbyr 120 private prosjekter, 200 sider på 2 minutter, men så mange sider man vil per kjøring, denne versjonen tilbyr og datalagring på DropBox eller S3 (parsehub, u.d.).

Rammeverket Scrapy er basert på Python og fungerer ved at man koder "web spiders" som søker opp sider og henter ut data ved hjelp av "selectors". Det er en tidkrevende prosess og tilpasse disse "web spiderene" til å kunne håndtere JavaScript, men de fungerer bra til å hente ut store mengder data (scrapy, u.d.).

Selenium WebDriver er et rammeverk fungerer med flere programmeringsspråk som Java, C# og Python. Dette rammeverket er designet for å drive med automatisert testing av nettsider og kommer derfor med naturlig støtte for JavaScript. Dette er et rammeverk som fungerer godt i små og enkle systemer der det ikke skal hentes ut store mengder data på en gang (selenium, 2021).

StormCrawler er et open-source software development kit (SDK) basert på Java, for å bygge søkeroboter basert på Apache Storm. Målet med StormCrawler er å bygge søkeroboter som er skalerbare, lett å utvide og fleksible. Denne nettskraperen krever at man har installert Apache Maven (stormcrawler, 2021).

2.8.4 Sentiment analyse

Det er mange verktøy for sentimentanalyse å finne, det består som oftest av komplette sentimentanalyseverktøy, eller ordbanker en kan bruke i sammenheng med et eget verktøy. Komplette verktøy er fordelaktige i at de eliminerer mye arbeid, men de er ofte veldig store, og dyre. Ordbanker eliminerer i stor grad prosessen med å definere ord, men de kan også komme i feil format og være mangelfulle.

Repustate leverte mer enn alt som var nødvendig for å oppnå et godt resultat for sentiment analyse av nyhetsartikler. Ved bruk av kunstig intelligens og maskinlæring for å oppnå bedre resultater. Direkte analyse av ord uten oversetting leder også til et mer nøyaktig resultat. Dessverre ligger denne løsningen priset til 299$ for standard plan, som ikke engang inkluderer norsk. Prisen er ukjent, men sannsynligvis dyrere for egendefinert plan (repusate data in sight, 2021).

Norec-FastText var en interessant løsning som baserer seg på å trene opp et program til å forstå sentimentet riktig. Den er gratis og open-source. Men dessverre var det vanskelig å finne noen dokumenterte resultater, samtidig som det ville kreves tid til å optimalisere den (Web64, 2019).

Afinn er en ordbank som er relativt god fordi den består både av en ordliste og rangering av ordene i form et vektsystem basert på hvor positivt eller negativt det er. Ulempen med Afinn

er at den i noen tilfeller vektlegger ordene på en dårlig måte, og det er en stor jobb å endre alle de manuelt. Samtidig har den relativt få ord, og det kan virke som at den ikke har de rette ordene en kikker etter i sammenheng med aksjehandel og artikler. (Hjertaker, 2018)

NorSentLex var enklere formatert enn Afinn, den viser ikke vektlagte ord. Denne har to lengre lister, en for positive og en for negative. Problemet med denne er at den har såpass mange ord i hver kategori at noen kan virke unormale. (Itgoslo, 2019)

2.8.5 Oppdaterte aksjepriser

For å holde seg oppdatert på aksjekursen til hvert et selskap finnes det mange nettsider som oppdateres ofte og man kan også få oppdateringer fra egne aksjemeglere som Nordnet. For å kunne finansiere sanntidsoppdatering av aksjekursen selger disse selskapene tjenestene sine i form av en API eller annonser på deres nettsider. Aksjemeglere har en annen måte å finansiere dette på ved hjelp av kjøp og salg av aksjer der de tar seg betalt for hver handel som er kalt kurtasje.

Yahoo finance er en del av yahoo sitt nettverk og tilbyr blant annet finansielle rapporter, pressemeldinger og sanntids aksjepriser. De tilbyr tjenester som API der man kan få tilgang til en database med diagrammer, pris, volum handlet og dato tilknyttet forskjellige aksjer.

Euronext tilbyr mye det samme som yahoo finance men har hovedfokus i europa og Oslo børs ble en del av Euronext i 2019. Dette tilsier at Euronext har mer hyppig oppdatering av den norske børsen i motsetning til yahoo men har mer begrenset tilgang til informasjon rundt deres API. For å få tilgang må du gjennom en søknadsprosess der kostnader ikke er offentliggjort som gjør det veldig vanskelig å sammenligne med andre API tjenester. (RapidAPI, 2021)

3 MATERIALER OG METODE

Dette kapittelet tar for seg hvilke programmer og verktøy som er brukt under prosjektet, valg av eksisterende løsninger og om testingen som er blitt gjort underveis i prosjektet

3.1 Utviklingsmiljø

Hvert medlem av utviklingsteamet hadde utviklingsmiljø de var mest kjente og komfortable med, og av den grunn har de følgende utviklingsmiljøene vært i bruk på tvers av medlemmene i teamet.

JetBrains tilbyr integrerte utviklingsmiljøer (IDE) for flere populære programmeringsspråk som Python, HTML, CSS, JavaScript og Java. IDEene de tilbyr kommer med utvidelser som autofullføring av kode og kompatibilitet med Git. I dette prosjektet har WebStorm, IntelliJ og PyCharm blitt brukt ettersom dette er IDEer som noen gruppemedlemmer hadde kjennskap til fra før. IDEene er i tillegg bygget på samme grafiske brukergrensesnitt slik at man som bruker kjenner til både **IntelliJ** og **PyCharm** ved å ha jobbet med **WebStorm**. **WebStorm** er JetBrains sin IDE for HTML, CSS og JavaScript, **IntelliJ** for Java og **PyCharm** for Python. (JetBrains, u.d.)

Netbeans er en open-source Java IDE, den tilbyr et enkelt brukergrensesnitt mens den også gir muligheter for diverse utvidelser en kan få bruk for. Hovedgrunnen til at vi har tatt i bruk Netbeans i prosjektet vårt til fordel for f.eks Intellij er at Netbeans har mye bedre støtte til å sette opp en lokal server der man kan laste opp koden for å teste. Samtidig kommer Netbeans med Maven støtte ut av boksen, så en trenger ikke å konfigurere den på noen som helst måte for å komme i gang. (Lamouchi, 2016)

Visual Studio Code er et «koderedigerings-program» det er ikke en IDE, men den kommer med mange kraftige funksjoner som IDEer ofte gjør. Eksempelvis er det «syntax highlighting, auto-fullføring og funksjonalitet til å teste koden. Tekst editoren kommer i utgangspunktet kun med støtte for JavaScript, TypeScript og Node.js, men har et bibliotek med masse utvidelser for de fleste programmeringsspråk. I dette prosjektet har Visual Studio Code hovedsakelig blitt brukt for HTML, CSS og JavaScript. Men også litt Python. (Visual Studio, u.d.)

3.2 Metode

I dette underkapittelet skal vi først ta for oss hvordan Scrum har blitt brukt i prosjektet før vi går over til verktøy som har blitt benyttet i prosjektet.

3.2.1 Scrum

Prosjektet har blitt utført ved bruk av den agile utviklingsmetoden Scrum hvor alle medlemmene i gruppen har jobbet tett og brukt programmet Jira til å opprettholde Scrum. Backloggen ble fylt med oppgaver ("stories") i løpet av planleggingsprosessen, men noen stories ble fjernet og andre lagt til i løpet av prosjektets levetid, da undersøkelser viste at noen av stories'ene ikke var gjennomførbare uten uakseptable økonomiske utgifter.

Det har blitt utført daglige Scrum møter i gruppen - hvor det har blitt diskutert framgang det siste døgnet, hva som skal utføres resten av dagen og om man trenger hjelp. Det har og vært praksis at de som er ferdige med oppgavene for sprinten har tilbudt seg å hjelpe til med de resterende oppgavene i sprinten.

Etter hver sprint, hver 14 dag har det blitt utført en sprint review med oppdragsgiver og veileder. I disse møtene har produktets funksjonalitet blitt presentert, og oppdragsgiver og veileder har kommet med mulige alternative løsninger som kunne føre til et bedre sluttprodukt. I tillegg var det oppdragsgiver som valgte hvor prosjektet skulle gå videre og sprintene ble planlagt med stor vekt på hans ønsker.

3.2.2 Verktøy

3.2.2.1 Jira

Det web-baserte programmet Jira er et program designet for Scrum-prosessen. Dette programmet inneholder muligheten til å lage stories, oppgaver og sprints, som igjen blir vist i Scrum-board og backlog. Programmet genererer også automatisk sprint-rapporter etter hver enkelt sprint. Utviklerteamet brukte dette programmet til både å planlegge og å utføre prosjektet på en smidig måte.

3.2.2.2 Kommunikasjonsverktøy

Utviklingsteamet kommunisert i hovedsak via plattformen **Discord**. Discord er et VoIP-program hvor man kan lage en dedikert server og invitere medlemmene som skal ha tilgang til serveren. Alle medlemmene på serveren kan se om det er noen som sitter i en samtale eller bli med i en samtale for å være tilgjengelig for andre medlemmer. I tillegg tillater dette programmet at man skriver på en felles samtale og deler skjerm til de som er aktive på

serveren. Utviklingsteamet lagde en server på Discord og brukte denne til å hjelpe hverandre med problemer, spørsmål og til Scrum møter.

Zoom er et VoIP-program med mulighet for videomøter og skjermdeling. Dette ble brukt ved Sprint review da utviklingsteamet hadde møte med oppdragsgiveren og veilederen siden dette gav muligheten til å "møtes" ansikt til ansikt.

3.2.2.3 Github

Utviklingsteamet brukte applikasjonen **GitHub Desktop** til å "hoppe" mellom grenene på prosjektet og til å hente og dytte de nyeste endringene gjort på prosjektet på GitHub. Nettsiden til GitHub ble brukt ved pull requests og code reviews for å se om to grener kunne kombineres eller ved å gjøre de nødvendige endringene for at grenene skulle kunne kombineres.

3.2.2.4 Confluence

Confluence er et web-basert program som består av flere individuelle verktøy. Retrospective, Product requirements (SRS) og Gliffy Diagram er noen få av de. I løpet av planleggingsprosessen brukte utviklingsteamet Gliffy Diagram til å lage **Wireframes** og diverse andre modeller for hvordan sluttproduktet skal se ut og fungere.

3.2.2.4 Microsoft Word og Microsoft 365

Microsoft 365 er et samleverktøy som inkluderer Microsoft Word. Word er et tekstbehandlingsprogram som utviklingsteamet har brukt til å skrive Bachelor rapporten. Microsoft 365 ble brukt sammen med Microsoft Word for å tillate flere brukere å skrive på det samme dokumentet samtidig, da det er enklere og holde kontroll på hva som er skrevet når alle ser det samme dokumentet i sanntid.

3.2.3 Analyse og valg av eksisterende løsninger

I dette prosjektet har vi sett på noen eksisterende teknologier for å komme fram til en løsning som passer dette prosjektet. Dette gjorde vi for å lære om de forskjellige teknologiene og for å prøve å komme fram til den mest effektive måten å utføre prosjektet på.

3.2.3.1 Aksjekurser

Vi har sett på forskjellige løsninger for å hente ut og vise aksjepriser for det norske aksjemarkedet i systemet. Her så vi på de to tilgjengelige tjenestene vi fant for å hente ut norske aksje priser. Planen om å benytte dette ble etter hvert vraket ettersom løsningene fra EURONEXT og Yahoo Finance ble for kostbare.Vi kunne bare hente ut 500 kall per aksje

per måned gratis. Etter 500 kall kostet det penger, og ble ugunstig for en eventuell implementering i prosjektet vårt. Dette ble besluttet på grunnlag av at det er rundt 250 selskaper på den norske børsen som gjør at vi kunne innhente informasjon bare 2 ganger per måned uten kostnad.

3.2.3.2 Loggbok

For loggboken så vi på eksisterende løsninger for å importere transaksjonslogg filer fra Nordnet og konvertere dem fra CSV til JSON. Ettersom CSV filen fra Nordnet kommer i TSV (tab-separated values) format endte vi opp med å bruke en eksisterende løsning for Excel filer som vi endret litt på slik at den fungerte for filene fra Nordnet. Vi kom fram til at denne løsningen fungerte bedre enn de eksisterende løsningene vi fant som var lagd for CSV filer.

3.2.3.3 Analyse

For å finne ut hvilke analyser vi skulle ha med og hvordan vi skulle framstille dem, så vi på forskjellige analyser av aksjer for å finne ut hva som kunne være relevant. Etter å ha laget noen forslag på analyser snakket vi med oppdragsgiver og veileder - som kom med ideer og innspill og lagde skisse over hvilke analyser de ønsket og hvordan de ønsket at de skulle se ut.

3.2.3.4 Nettskraper

Ingen i gruppen hadde erfaring med nettskrapere, men vi måtte lage en egen skraper fra bunnen av, da de ferdige løsningene vi fant ikke hadde all funksjonaliteten vi trengte uten å måtte betale for det. Av den grunn måtte vi undersøke forskjellige rammeverk før vi bestemte oss for hvilken vi ville bruke. Vi så på rammeverkene Scrapy, StormCrawler og Selenium WebDriver. StormCrawler ble ikke valgt, da dette rammeverket er bygget på Java og vi hadde planlagt å bruke Python. Oppgaven til nettskraperen var repetativ og begrenset. Så i valget mellom Scrapy og Selenium WebDriver, valgte vi Selenium for å holde prosjektet enkelt - og om nettskraperen skulle utvides til nettsider som utnyttet seg mye av JavaScript, så var støtte for dette allerede innebygget.

3.2.3.5 Sentiment

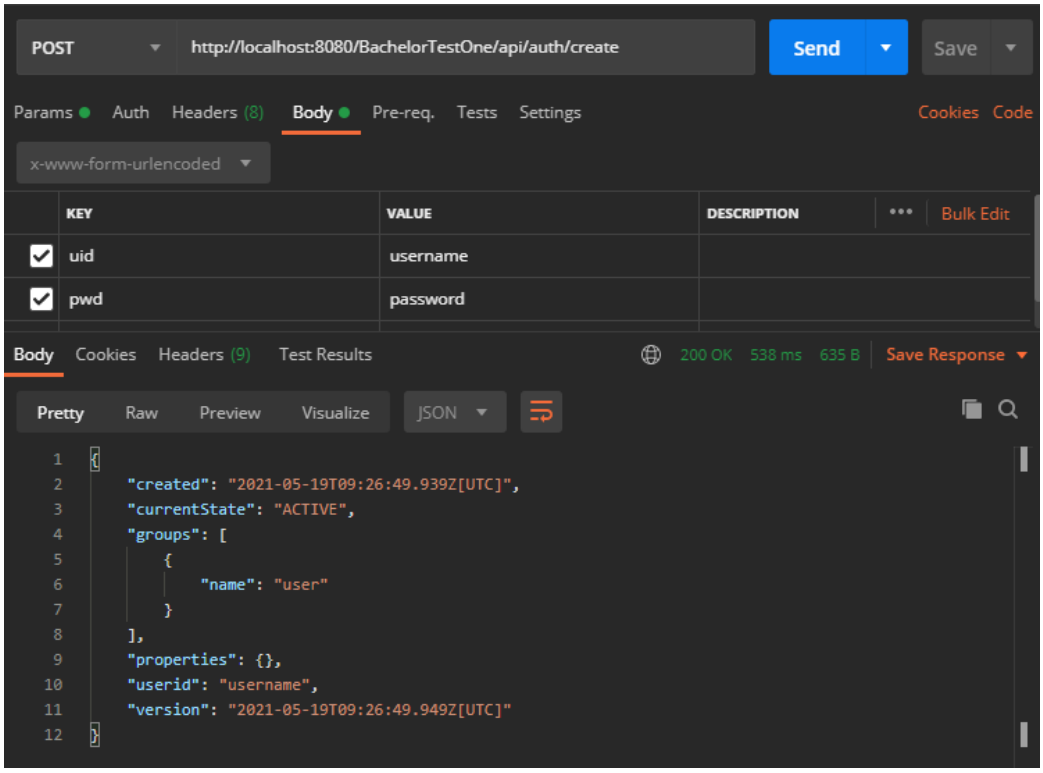
Det er en del sentiment-analyseverktøy å finne på nett. Men kravene vi hadde var at det skulle være gratis, det skulle passe til artikler som omhandler aksjer, og fungere med norsk språk. Etter å ha lett litt uten å finne verktøy som tilfredsstiller kravene vi hadde, så bestemte vi oss for å utvikle vår egen analyse ved hjelp av en ord-pakke vi fant på GitHub og

modifiserte til å løse våre behov (Itgoslo, 2019). Denne ordbanken består av ca 20 000 ord og er delt opp i negative og positive ord. En del av disse ordene er ikke nødvendigvis i noen av kategoriene, og har derfor blitt fjernet.

3.2.4 Testing av funksjonalitet

Vi har ikke brukt noen automatiske tester, men testet programmets funksjonalitet manuelt underveis og med jevne mellomrom gått tilbake og testet funksjonalitet som har blitt implementert. For loggboken lagde vi dummy data i CSV filer som vi lastet opp og sjekket at filopplasting og transaksjonsloggen fungerer. Denne dataen ble så brukt for å sjekke at analysene fungerte ettersom de bruker data transaksjonsloggen for å gjøre analysene. For sentimentet analysen sjekket vi resultatet og sjekket at det stemte overens med artikkelens innhold.

For å teste webserveren har vi tatt i bruk Postman. Figur 2. Det lot oss teste APIen til serveren uten å måtte manuelt skrive JavaScript kode. I Postman programmerte vi funksjoner som lagret informasjon i et miljø. Deretter kunne vi hente ut informasjonen i andre forespørsler. Dette tillot en høy grad av automatisering av testing. Ved å bruke Postman til å teste kunne vi selektivt teste de komponentene det var behov for, før vi skrev noe front-end kode, som førte til effektivisering av programmeringen.



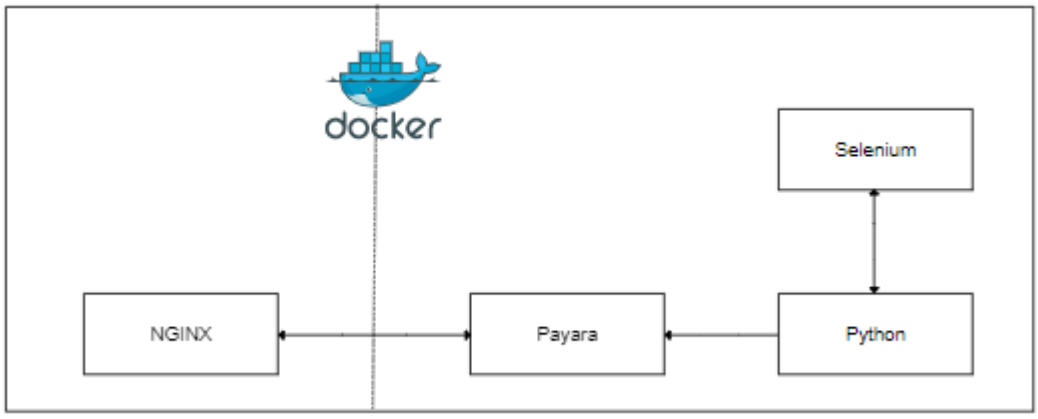
Figur 2 - API respons i Postman

4 RESULTATER

I dette kapittelet vil det bli beskrevet hvordan de individuelle elementene i systemet fungerer, samt hvordan det komplette systemet er bygget opp.

4.1 Oversikt over systemet

Dette prosjektet består av fire Docker "kontainere", der NGINX inneholder websiden, Payara holder webserveren, Python kontaineren kjører skriptene og utgjør nettskraperen som tar i bruk Selenium for å navigere på web. Figur 3



Figur 3 - Oversikt over Docker kontainere

Funksjonaliteten i prosjektet er i hovedsak bygget opp av tre individuelle elementer som kan tilpasse og modifiseres uten spesiell innvirkning på hverandre.

1. Loggbok med analyse
2. Nettskraper
3. Sentimentanalyse

4.1.1 Funksjonalitet

Loggboken fungerer ved at man laster opp en transaksjonslogg fra Nordnet, som blir konvertert fra CSV til JSON format og sendt til sendt serveren. Når transaksjonsloggen lastes opp til serveren, så blir den delt opp og relevante data blir plassert inn i en tabell i databasen, mens ikke-relevante data blir filtrert bort. Neste gang man da går inn på transaksjonsloggen i nettleseren, så sender nettleseren en "Get request" (via et JavaScript med en "JSON web token") som header i requesten. Når serveren mottar denne "Get requesten" med en gyldig token, så henter den ut transaksjonsloggen fra tabellen den ligger i og sender denne til nettleseren, som igjen sorterer og presenterer dataene for brukeren.

Nettskraperen blir starter på det samme tidspunktet hver dag ved hjelp av Cron som er implementert i Docker "konteineren" og fungerer ved å hente en liste med aksjer fra serveren, navigere seg inn på en nettside, for så å søke etter hver enkelt aksje. Deretter går nettskraperen inn på flere av de artiklene som dukker opp og henter ut innholdet. Så sender nettskraperen alt innholdet til sentiment analyse skriptet. Dette skriptet traverserer deretter innholdet og ser etter positive og negative ord, før den gir en karaktervurdering som den returnerer til nettskraperen, sammen med en liste over positive og negative ord som er funnet. Når nettskraperen mottar dette, så lagrer den resultatet i en array, før den gjentar prosessen til den er ferdig. Deretter sender nettskraperen den komplette arrayen til databasen som igjen deler opp denne informasjonen og legger det inn i en tabell.

4.1.2 Design

Gruppen valgte å gjenbruke et lett tilpasselig design av nettsiden fra et tidligere prosjekt fra en av medlemmene i gruppen, slik at det skulle være mer tid til å lage funksjonalitet. Hver side av nettsiden har blitt tilpasset sitt bruksområdet, og blitt modifisert med en linje på toppen av siden for en RSS-feed.

4.1.2.1 Don Normans 6 brukbarhets prinsipp

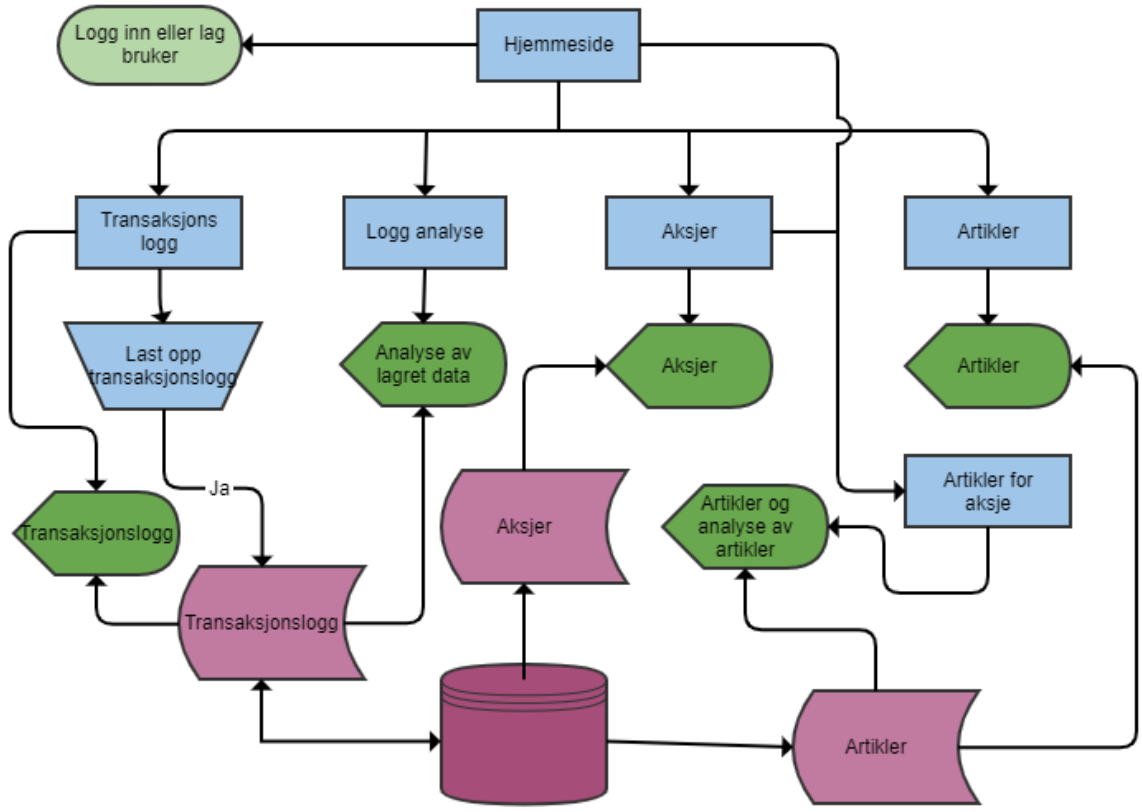
Generelt så følger systemet de 6 prinsippene fra Don Norman, 2.3.1 Don Normans 6 brukbarhets prinsipp. Funksjonene i systemet gir visuell feedback i form av at nettsiden vil endre seg når brukeren aktiverer funksjonene. Funksjonaliteter som er kodet inn i sidene er de som er ansett som nødvendige. Det er god mapping i systemet og funksjonene er bygget opp likt for lik funksjonalitet. Navigasjonsfeltet i systemet åpnes ved å trykke på den velkjente meny knappen som vist i Figur 4, og for å lukke den trykker man på en X.

Figur 4 - Ikon for hamburgermeny

4.1.2.2 Objektorientert design

I løpet av prosjektets levetid har det blitt lagt vekt på SOLID og DRY prinsippene og i all hovedsak er disse fulgt. Det finnes noen få unntak som for eksempel duplikat HTML og JavaScript kode. I et forsøk på å unngå dette så bruker de fleste sidene en felles CSS, og navigasjonsfeltet blir skrevet inn i nettsiden av et JavaScript når man åpner sidene, 2.3.2 Objektorientert design.

4.2 Frontend

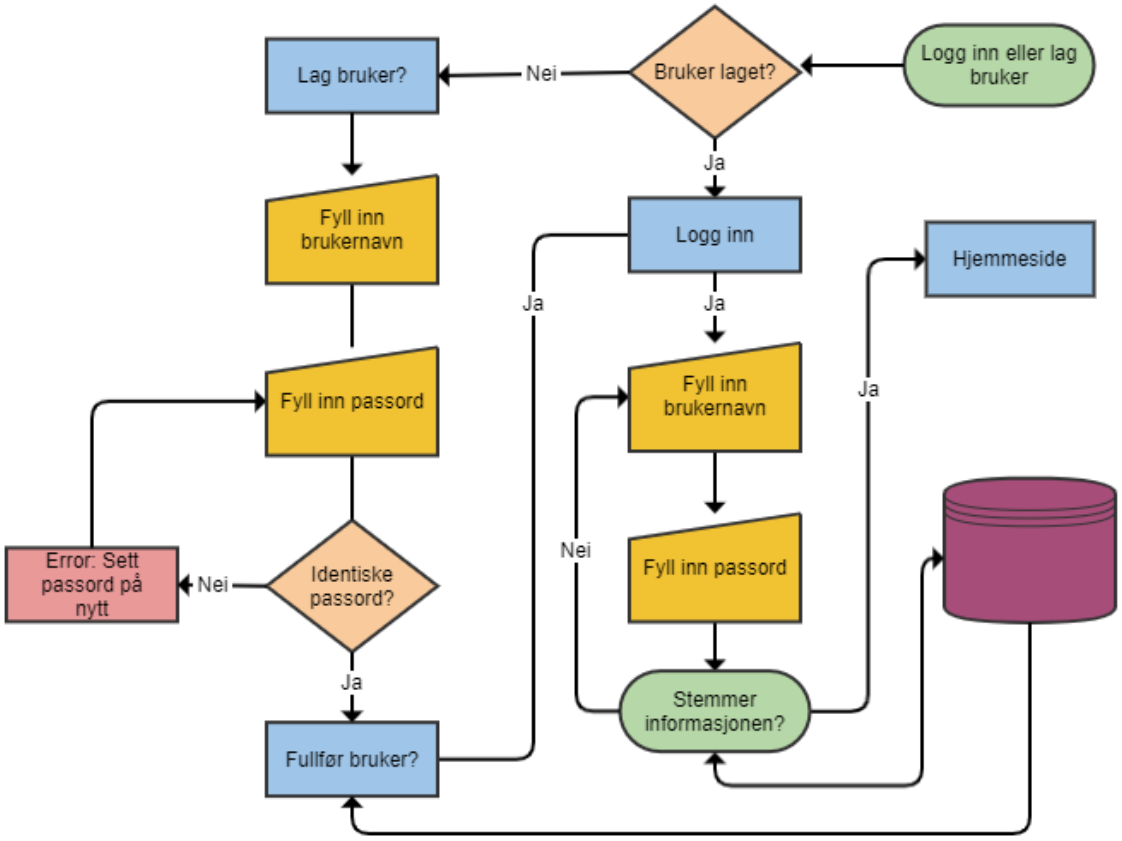


Figur 5 - Flow diagram av komponenter

Figur 5 er et flow diagram som viser hvordan systemet er bygget opp, hvordan man kan traversere systemet i nettleseren, hva som blir vist fram og analysert.

Dette kapittelet skal forklare hvordan brukersiden av systemet er bygget opp og hvordan den fungerer ved å gå inn i hver enkelt side av det web-baserte systemet.

4.2.1 Login



Figur 6 - Flow diagram for autorisering

For å logge på må du lage bruker på innloggingssiden til “prosjektet” “link”. der kan du lage en bruker ved hjelp av en gyldig e-postadresse og et passord som enkle sikkerhetskrav.

Login trenger to gyldige verdier for å autorisere en bruker inne på serveren. Det ene er en validert email og det andre er et gyldig passord. Dette lagres når en bruker lager en ny bruker, se Figur 6. Da er det satt noen passordkrav med javascript for å forhindre at passordet blir en sikkerhetsrisiko for brukeren. Det kommer også visuelle notifikasjoner hvis bruker ikke oppfyller kravene og forhindrer innsendelse til serveren. Hvis alle krav er oppfylt vil informasjonen sendes til serveren som da vil sende en verifisering email til den angitte e-postadressen. Når det blir sendt en verifisering tilbake til serveren vil email og passord lagres i databasen.

Hvis passord er blitt glemt kan du trykke på en link på innloggingssiden, der kan man skrive inn epost-adressen man brukte til å lage bruker, og via epost får man et nytt automatisk generert passord, med en oppfordring om å manuelt endre passord.

4.2.2 Hjemmeside

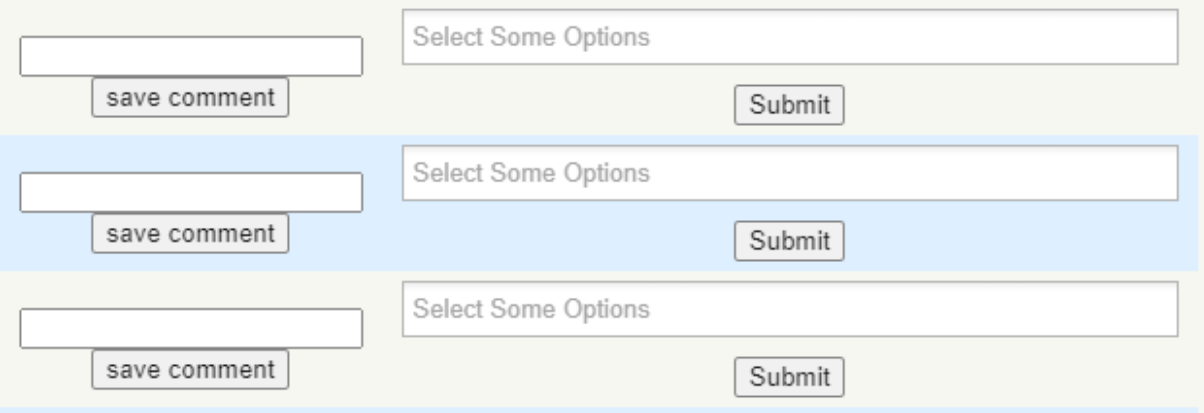
På hjemmesiden av prosjektet er det et felt som beskriver prosjektet og et felt som viser de fem mest omtalte aksjene den dagen og den siste måneden. Dette feltet viser hvilke aksjer som er mest populære og hvor mange artikler som er publisert i den perioden. Om man da trykker på et av aksje navnene blir man omdirigert til en side hvor alle artiklene som er samlet inn om den spesifikke aksjen blir presentert, 4.2.7 Artikler per aksje.

Når man besøker hjemmesiden, vil denne siden bare bestå av feltet med informasjon om prosjektet. For å få full utnyttelse av siden så må man logge inn, dette gjør man ved å trykke på knappen som vises i øvre venstre hjørnet, Figur 4. Om man allerede er innlogget vil denne knappen se ut som, og man vil kunne utnytte all funksjonaliteten på siden.

4.2.3 Transaksjonslogg

På denne siden kan brukeren laste opp og se transaksjonsloggen sin. For å laste opp transaksjonslogg trykker man “velg fil”, dette lar brukeren velge hvilken CSV fil de vil laste opp. Når man har valgt fil trykker man på “last opp”. Programmet vil da lese filen som en binær streng, endre alle komma til punktum før den bruker JavaScript biblioteket XLSX til å gjøre den binære strengen om til et JSON array som deretter sendes til server.

Under dette kan brukeren se kjøp og salg transaksjonene som har blitt lastet opp. Dette gjøres ved at programmet gjør et kall til serveren og mottar transaksjonsloggen til brukeren. Når loggen mottas går den gjennom en løkke som lager en tabell som skrives på HTML siden. For hver transaksjon legges det også inn et tekstfelt for å legge inn personlig notat for hver transaksjon med en tilhørende lagre knapp for å lagre kommentaren. På hver transaksjon lages det en dropdown meny, denne menyen bruker JavaScript plugin Chosen som er en plugin for å lage dropdown menyer med forms. I denne menyen kan brukeren velge flere tags ut fra en liste som de mener passer til transaksjonen, se Figur 7. Når disse er valgt trykker man på submit for å lagre dem, tags som er lagret til transaksjonene blir brukt i analysen.



Figur 7 - Kommentar og Tag

4.2.4 Logg analyse

På denne siden kan brukeren se en rekke analyser over salgene fra transaksjonsloggen. Siden henter inn den opplastede transaksjonsloggen til brukeren og finner ut hvilke transaksjoner som er salg før den utfører analysene. Når analysene er utført plottes dataen ved hjelp av Javascript biblioteket Plotly for å vise analysene grafisk på siden.

Den første analysen er Win rate som viser i prosent hvor stor andel av salgene som er solgt med fortjeneste. Dette finner den ut ved at den går igjennom en liste med alle salgene og teller opp hvor mange av dem som har fortjeneste, før den deler totalen med 100 og ganger med antallet som har fortjeneste, deretter plottes dette svaret inn i grafen.

Den andre analysen er en Profit and loss(PNL) analyse over tid. Denne grafen viser når salgene er utført og resultatet fra hvert enkelt salg. Dette gjøres ved at programmet går gjennom listen over salg og oppretter to lister. En liste for datoene til alle salgene og en liste for resultatet av salgene, disse listene brukes så for å plotte PNL diagrammet, der listen med datoer fungerer som x-koordinater og listen med resultater fungerer som y-koordinater.

Den tredje analysen er en Tag analyse som viser en candlestick graf over tags som er i bruk i transaksjonsloggen. Dette gjøres ved at programmet går igjennom salgslisten og finner alle salg med en spesifikk tag før den finner min, maks, median, øvre og nedre kvartil verdiene for denne taggen. Dette gjøres for alle tags som eksisterer i listen, og når den er ferdig plottes dataen den har funnet inn i en graf. Den fjerde analysen er Stock analyse som fungerer på samme måte som Tag analyse, men for aksjer istedenfor tags.

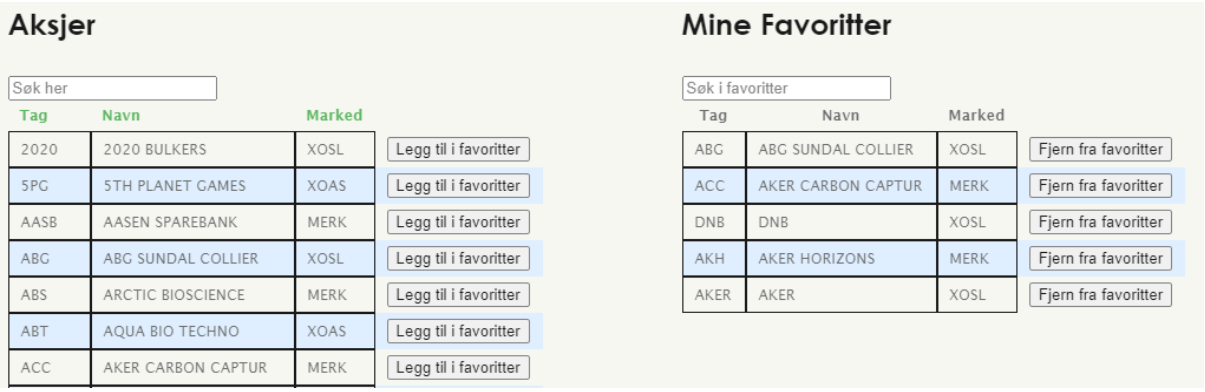
Den siste analysedelen er en single tag analyse hvor brukeren først velger hvilken tag de vil se analyse for. For å velge tag kan brukeren enten velge fra en dropdown meny eller trykke på søylen i tag analyse for den taggen de vil se single tag analyse for. Under dropdown

menyen kan man se en gjennomsnittlig PNL for taggen i prosent. Under dette listes alle salg med denne taggen med dato for salget, navnet på aksjen og resultatet i prosent med en grafisk horisontal søyle som er grønn og til høyre for resultatet om resultatet er profitt. Om resultatet ikke er profitt vil søylen være rød og til venstre for resultatet. Under denne listen vises to lister over tag kombinasjoner, hvor den en viser de beste tag kombinasjonen med den valgte tagen og den andre lister opp de verste tag kombinasjonene med den valgte tagen. For å finne disse kombinasjonen går programmet gjennom listen som fant salgene for den valgte tagen og finner alle salg med flere tags. Deretter regner den ut det gjennomsnittlige resultater for hver kombinasjon. Så sorteres denne listen etter størrelsen på resultatet. For listen som viser de beste kombinasjonene printes de 5 største i listen og for verste kombinasjoner printes de 5 laveste resultatene.

4.2.5 Aksjer

På venstre siden er en liste over tilgjengelige aksjer. Aksjene fra listen kan legges til i favoritter som listes opp på høyre side. Over begge listene har man en søkelinje for å søke i de forskjellige listene. Brukeren kan trykke på navnet på en aksje for å gå til en side som viser alle tilgjengelige nyhetsartikler for den spesifikke aksjen med sentimentanalyse

På denne siden vises to lister. En liste for alle tilgjengelige aksjer og en liste for favorittaksjer. Figur 8. Når siden åpnes gjøres det to kall til serveren en for å hente ut tilgjengelige aksjer og et kall for å hente ut favorittaksjer. Så looper programmet gjennom disse listene og printer dem ut på HTML siden. Ved siden av aksjene på tilgjengelige aksjer er det en knapp for å legge til den spesifikke aksjen til favorittlisten og favoritt listen har tilsvarende knapper for å fjerne dem fra listen. Alle aksje navnene i begge listene er interaktive og kan trykkes på for å bli sendt videre til artikkel per aksje siden for den spesifikke aksjen. (se 4.2.7 Artikler per aksje)



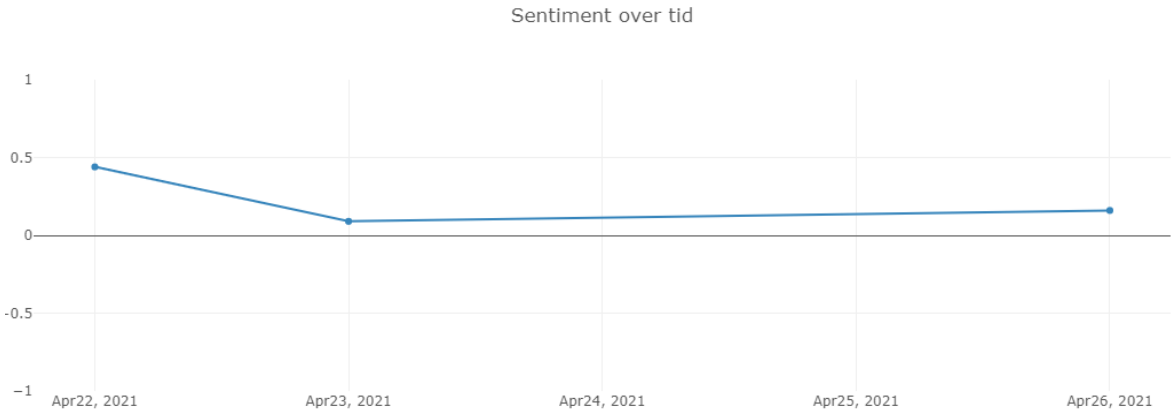
Figur 8 - Tilgjenglige og Favorittaksjer

4.2.6 Artikler

På artikkel siden vises de 40 siste artiklene som er lagt ut på nettavisene det innhentes informasjon fra. Her hentes datoen den ble publisert, tittelen med hurtiglink til artikkelsiden, hvilket selskap det omhandler og en positiv, negativ eller nøytral indikator som blir bestemt av en analyse, se 4.3.7 Artikkel sentiment. Grunnlaget for å ha en link til artikler er at bruker selv kan se hva artikkelen omhandler og gjøre sin egen formening uavhengig av vurdering analysen har gitt. Denne siden fungerer ved å sende en "Get request" til serveren, hvor serveren sender alle artiklene lagret i databasen som respons. Deretter blir responsen sortert fra nyeste til eldste artikkel og de 40 første artiklene i listen blir presentert i en tabell. Figur 9.

Figur 9 - Tilgjengelige artikler

4.2.7 Artikler per aksje



Figur 10 - Sentiment over tid graf

Denne siden viser informasjon tilknyttet spesifikke aksjer, hvilken aksje blir valgt enten på hjemmesiden, 4.2.2 Hjemmeside, eller på aksjesiden, 4.2.5 Aksjer. På toppen av denne siden er det en graf som viser sentiment over tid. Figur 10. Da denne ble laget var det en utfordring med at flere artikler på en dag gjorde denne grafen temmelig uleselig. Dette ble løst ved å ta gjennomsnitts sentimentet for alle artikler som ble publisert på en spesifikk dag og sette denne som x verdi. Øvre og nedre grense på hvor høyt og lavt sentimentet kunne være ble og låst for å unngå ekstreme verdier som igjen ville gjort grafen vanskelig å tyde. For å løse dette blir sentiment vurderingen sjekket om den er mellom -50 og 50 før den blir delt på 50 og satt som y verdi, om vurderingen er utenfor disse verdiene blir y verdien satt som 1, deretter blir disse verdiene plottet inn i grafen. Denne siden viser og en liste på maks 40 artikler skrevet om denne aksjen, da det bør være en begrensning på hvor mange elementer som kan dukke opp.

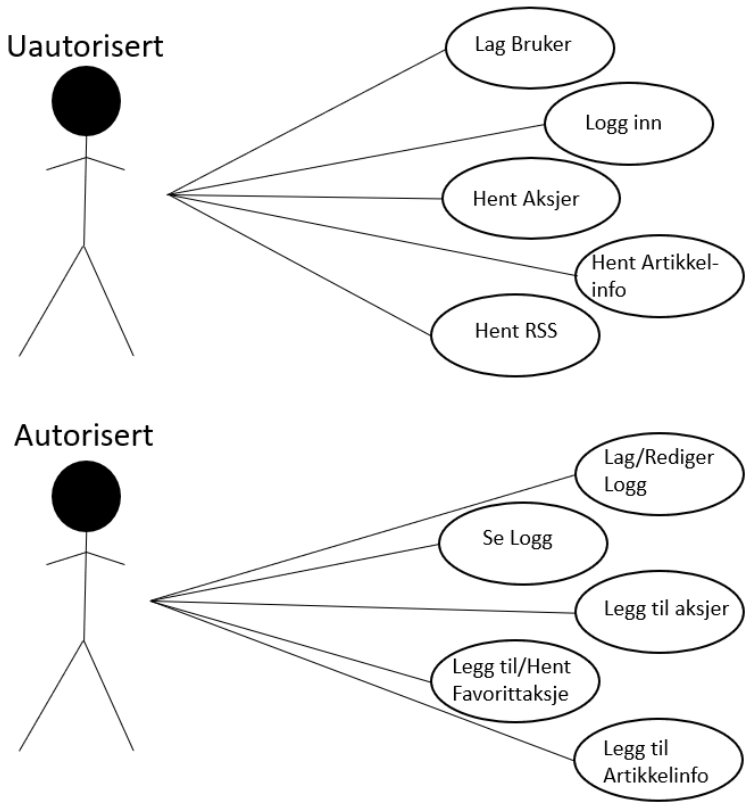
4.3 Backend

Backend består av en webserver i form av Java EE 8 som er opplastet på et Payara Docker image. Og et Python script som crawler en nyhetsportal og sender artikkelinformasjon og sentiment til webserveren, automatisert ved hjelp av Cron. For å koble alt dette sammen til et enkeltstående system ble det skrevet en Docker-compose fil, som videre tillot enkel deployment av hele systemet.

4.3.1 Webserver

Webserveren er en Java EE 8 implementasjon som tar i bruk mikrotjenester for å skille hver modul. Den er delt opp i en pakke per hovedfunksjon den har. Det eksisterer en JAX-RS konfigurasjons klasse som brukes til å fortelle Java EE hvilke pakker som skal eksponeres

som API. Det er også en sikkerhets konfigurasjons klasse som sørger for at en bruker som skal aksessere databasen er klarert. Og en kilde konfigurasjons klasse som definerer hvor, og hvilken type database som blir brukt. Oversikt over tillatelser for autoriserte brukere og uautoriserte brukere vises i Figur 11.



Figur 11 - Use Case diagram for Webserver funksjonalitet

Autoriseringssystemet består i stor grad av kode fra en eksempeloppgave av lærer i applikasjonsutvikling, videre brukt i eksamensbesvarelse i applikasjonsutvikling. Men noe er modifisert til å passe til oppgaven vår, samtidig er noen konfigurasjonsfiler brukt basert på disse.

https://bitbucket.org/mikaelntnu/chatserver/src/master/src/main/java/no/ntnu/tollefsen/auth/

https://github.com/Nitram2441/ExamProject/tree/master/src/main/java/app/examproject/auth

Autorisasjon Systemet består av klasser som definerer en bruker, og en gruppe. Videre følger en autoriserings service klasse som tar imot alle forespørsler som omhandler brukere, f.eks logg inn, lage bruker m.m. Denne klassen er avhengig av en nøkkel service klasse, som utfører alle operasjoner relatert til JSON Web Token. Videre har man en klasse som

konstruerer gruppene ved oppstart, dersom de ikke allerede eksisterer, og en klasse som sender mail til brukeren dersom den ønsker å skifte passord. Her hentes innloggingsinformasjonen til mail servicen fra en konfigurasjon klasse.

Loggføringsystemet består av en entitetsklasse som holder på informasjon om et enkelt logg-element, og en service klasse som behandler disse elementene og forespørsler i relasjon til dem. Service klassen får forespørsler om å legge inn nye elementer i databasen i form av en JSON fil, den filtrerer ut den viktige informasjonen oppretter objekter og lagrer de i databasen. Videre har den funksjonalitet til å legge til tags, og legge til kommentarer. Det opprettes automatisk tag på alle handler, som er dagen handelen ble utført på.

Delen som behandler individuelle aksjer består av en entitetsklasse og en service klasse. Entitetsklassen består av marked, navn, tag (som er aksje navnets forkortelse) og en id som genereres av serviceklassen basert på marked og tag. Serviceklassen mottar en liste i form av JSON, henter ut informasjon, lager objekter og legger de inn i databasen. Videre tillater den å hente ut alle aksjer, eller spesifikke aksjer.

Som en utvidelse til aksjer har man en pakke som sørger for å opprette brukeres favorittaksjer. Denne består av en entitetsklasse og en service klasse. Entitetsklassen genererer en id automatisk, og den inneholder også aksje-id og bruker id. Serviceklassen tillater å legge til favorittaksjer ved hjelp av aksje-id sendt som form-data og brukerens token som identifikasjon. Den tillater også å slette favorittaksjer ved hjelp av id som form-data, og å liste opp alle brukerens favorittaksjer.

Artikkelinformasjon består også av en entitetsklasse og en serviceklasse. Entitetsklassen består av informasjon om artiklen, som navn og URL. Den inneholder også en sentiment rangering, og to lister med positive og negative ord. Serviceklassen tar i bruk metoder for å motta og behandle JSON filen for så å legge inn entitetene i databasen. Spesielt for denne klassen er at den har en metode som finner siste dato for artikkelinformasjon i databasen, dette brukes i nettskraperen for å fastslå hvilke artikler som er for gamle og dermed kan ignoreres.

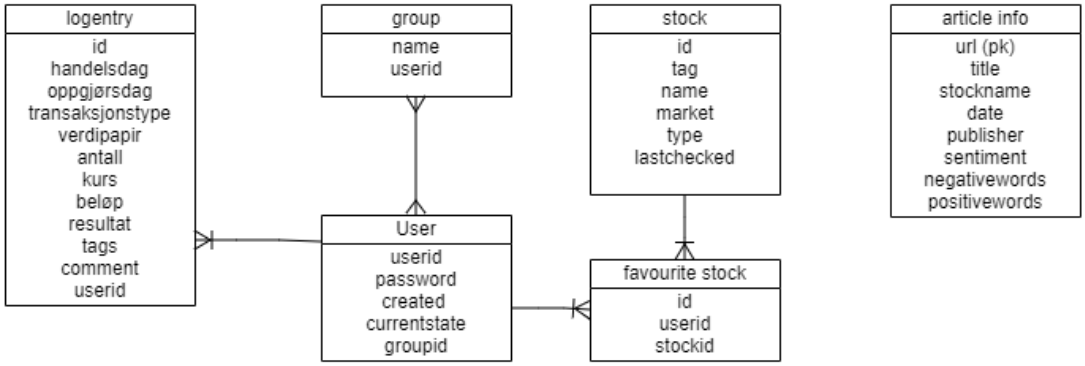
4.3.1.1 RSS

Som en del av web-serveren ble det bygget en RSS-leser. Denne RSS-leseren fungerer ved å bli aktivert av en “Get request” fra et JavaScript på nettsiden, og når dette skriptet blir aktivert så åpnes det en TCP forbindelse opp mot RSS-feeden og sender en “Get request”. Når RSS-feeden mottar og godtar denne requesten, så svarer den ved å sende hele feeden som

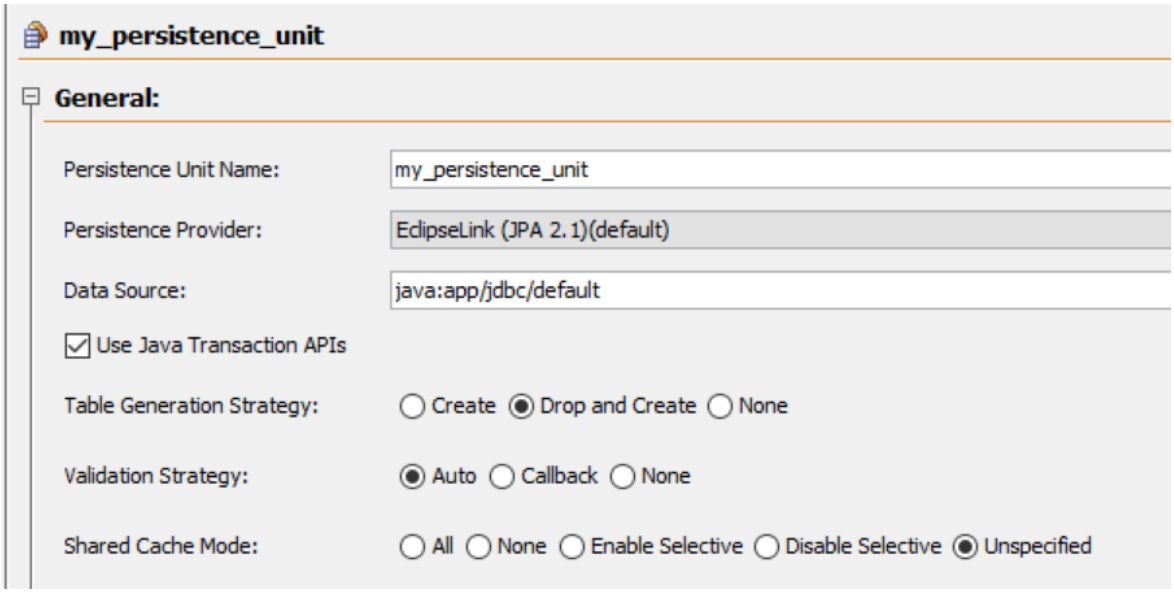
respons. Deretter blir responsen endret til en String i RSS-leseren, før TCP forbindelsen blir avsluttet. Stringen er på dette punktet en lang tekst som inneholder flere forskjellige elementer, denne blir så delt opp i en liste med elementer. Det første elementet i listen blir ignorert mens de neste fem elementene blir lest gjennom av skriptet og relevant data som tittel, lenke og bildet blir hentet ut og lagt til i en ny String. Denne strengen sendes så som respons til "Get forespørselen" som aktiverte skriptet.

4.3.2 Database

Det har blitt tatt i bruk Payaras innebygde default databaseløsning H2 i dette prosjektet. Databasen lagrer på all informasjonen som brukes i henhold til diagrammet under. Figur 12. Databasen er satt opp av webserveren, og all manipulasjon skjer via den. Webserveren bruker en "Persistence" fil av XML format til å definere hvordan den skal finne databasen, samt hvordan den skal samhandle med databasen. Netbeans kommer med en ui for å konfigurere denne filen. Figur 13.



Figur 12 - Oversikt over databasetabeller



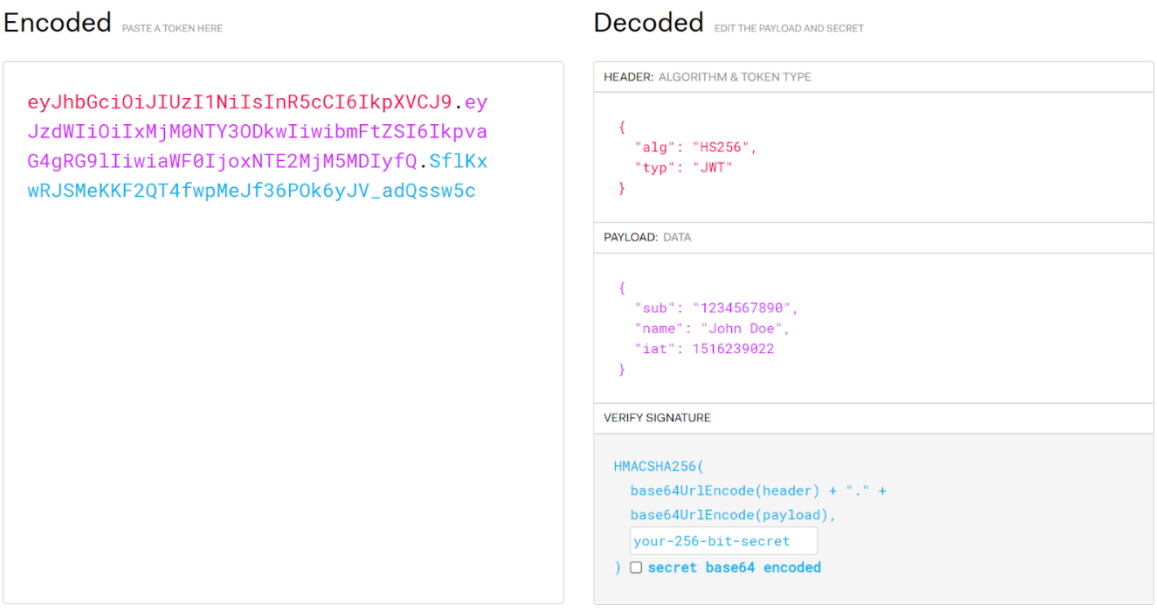
Figur 13 - Netbeans ui for databasekonfigurasjon

4.3.3 Brukerhåndtering og autorisering

Når man lager bruker på nettsiden, vil brukernavn og passord lagres i en database der passordet blir kryptert. Figur 14. Når en logger inn i systemet kommer en token i response, denne brukes i header ved ytterligere forespørsler til serveren. Server bruker token til å identifisere bruker, og for å validere hvilket innhold bruker er klarert til å se på. For å lage tokens er JSON Web Token systemet tatt i bruk, det denne gjør er å ta data sendt fra klient og kryptere den ved bruk av JSON objekt av body og header, samtidig som den bruker nøkler for å opprette en tekststreng som blir token. Figur 15. (JWT, u.d.)

```
user = new User();
user.setUserid(uid);
user.setPassword(hasher.generate(pwd.toCharArray()));
Group usergroup = em.find(Group.class, Group.USER);
user.getGroups().add(usergroup);
```

Figur 14 - Kode for oppretting av bruker



Figur 15 - Visualisering av oppbygging av token

4.3.4 Databehandling

Når brukeren laster opp en CSV fil fra Nordnet med brukerens transaksjoner vil disse først bli omgjort til JSON for å bli sendt til serveren. På serveren vil de filtreres og kun nyttig informasjon hentes ut og lagres i databasen. Når det videre hentes ut fra serveren vil hele elementet sendes, men transaksjoner som er unyttige (f.eks innskudd og renter) vil filtreres bort.

4.3.6 Nettskraper

For innsamling av artikler ble det først bygget en søkerobot ved hjelp av Selenium WebDriver i Python. Selenium WebDriver er et open-source rammeverk som tillater at man bruker og styrer en ChromeDriver. ChromeDriver er et verktøy som kan styre Chrome nettleseren uten et grafisk brukergrensesnitt. Når vi igjen skulle samle inn data til sentimentanalyse så ble denne søkeroboten bygget om til en nettskraper.

Denne nettskraperen henter ut en liste med aksjer og en dato for forrige gang den kjørte fra databasen. Deretter, lages et objekt med XPathene til de elementene nettskraperen skal samhandle med, før dem så navigerer seg inn på sidene til Dagens Næringsliv (DN). Her bruker nettskraperen objektet til å lokalisere innloggingsknappen som den trykker på, finner brukernavn og passord feltene og fyller inn disse før den trykker på logg inn. Deretter blir nettskraperen omdirigert til hjemmesiden til DN, hvor den igjen bruker objektet til å finne søkefeltet og fylle inn dette med en av aksje navnene før den trykker på søkeknappen. Her

bruker nettskraperen objektet til å finne feltet med antall søkeresultater, om dette viser 0 artikler går nettskraperen tilbake til hjemmesiden og søker på en ny aksje.

Om dette feltet viser 1 eller flere artikler så tester nettskraperen datoen artikkelen ble publisert opp mot datoen for forrige gang skriptet kjørte. Om artikkelen ble publisert før skriptet ble kjørt forrige gang så navigerer nettskraperen seg tilbake til hjemmesiden, om artikkelen er nyere så lager nettskraperen et nytt objekt med aksje navn, tittel, dato, lenke og utgiver. Deretter trykker nettskraperen seg inn på artikkelen hvor den henter ut innholdet og sender det til sentiment analysen. Sentiment analysen returnerer en sentiment vurdering, positive og negative ord, dette blir da lagt til i objektet med artikkelinformasjon. Deretter blir artikkel objektet sendt til en funksjon som bruker objektet til å lage et JSON objekt som blir lagt til i en JSON-array.

Nettskraperen går så tilbake til søkeresultatet og tester om neste artikkel i listen er nyere eller eldre. Deretter lager den et nytt objekt med informasjonen og sentimentet til denne artikkelen og lager et JSON-objekt av dette objektet og legger det til i JSON-arrayen. Dette gjør nettskraperen fram til den har hentet fem artikler eller frem til artikkelen er for gammel. Så går nettskraperen over til neste aksje på liste og utfører de samme handlingen for artiklene til denne aksjen og. Når nettskraperen har søkt på og hentet inn artikler for alle aksjene så sender nettskraperen JSON-arrayen den har laget til serveren som legger inn hvert JSON-objekt fra JSON-arrayen inn i en tabell i databasen.

4.3.7 Artikkel sentiment

Artikkel sentimentet er et skript som ble laget som et “Proof of Concept”. Dette skriptet samhandler med nettskraperen, ved at nettskraperen sender innholdet fra en artikkel til skriptet. Skriptet deler så innholdet opp i enkeltord hvor det fjerner alt utenom bokstaver. Deretter går skriptet gjennom listen ord for ord og tester de opp mot en liste med negative ord og en liste med positive ord. Om ordet blir funnet i den positive listen så blir ordet plassert i en liste med positive ord som er funnet og den totale sentiment vurderingen får pluss 1, om ordet blir funnet i den negative listen så blir ordet lagt til i en liste med negative ord som er funnet og den totale sentiment vurderingen får minus 1. Når hele innholdet har blitt testet opp mot listene så blir den totale sentiment vurderingen, listen med positive ord og negative ord funnet returnert til nettskraperen.

5 DRØFTING

I dette kapittelet skal vi drøfte de tekniske resultatene hvor vi går igjennom løsninger som er gjort i prosjektet og knytter dem opp mot teori og metode før vi går videre til utviklingsprosessen av prosjektet.

5.1 Drøfting av tekniske resultater

I dette underkapittelet skal vi drøfte de tekniske resultatene for frontend og backend i prosjektet. her vil vi drøfte om løsningene vi har valgt stemmer med teori og metode og om det er noen mulige forbedringer for prosjektet.

5.1.1 Frontend

I dette prosjektet valgte vi å bruke en felles CSS. Dette gjorde at vi kunne ha en lik stil over hele nettsiden uten å duplisere CSS kode for hver side. Fordelen med dette er at det er enkelt å endre stilen på alle nettsidene og å få de til å se like ut. Utfordringen med å bruke en felles CSS er at om du endret noe i CSS, kunne du risikere å endre noe på en av de andre sidene, men dette ble lettere utover prosjektets varighet. Etterhvert flyttet vi navigasjonsfeltet over til et script for å ikke ha duplisert kode. Dette gjør også at ved senere utvidelser, så trenger man kun å endre/legge til i det ene scriptet. Vi vurderte også PHP som en mulighet, men ettersom ingen i prosjektet hadde erfaring med PHP, så valgte vi å bruke JavaScript for å sikre kvalitet og gjennomføringsevne.

5.1.1.1 Don Normans 6 brukbarhets prinsipp

Generelt så følges disse prinsippene i systemet, med noen unntak.

Man får ikke visuell tilbakemelding når man skal laste opp transaksjonsloggen fra Nordnet. Når man aktiverer denne funksjonen må man også laste inn siden på nytt. Dette bryter Don Normans prinsipp om tilbakemelding. En mulig løsning på dette kunne vært at siden hadde lastet inn på nytt automatisk ved opplastning.

Et par av funksjonene burde blitt mer synliggjort da de ser ut som normal tekst, men når man drar musepekeren over denne teksten så vil den endre seg til en link pointer. Dette er en svak implementasjon av Don Normans prinsipp om synlighet. Ytterligere synliggjøring kunne blitt oppnådd ved å endre skrifttype eller utheving av skriften med en farge.

5.1.1.2 Transaksjonslogg

Et av kravene til prosjektet var å lage en side hvor man kan laste opp transaksjonsloggen fra Nordnet og ha mulighet til å se de opplastede transaksjonene. Her skulle det være mulig å

legge inn personlige notater for å dokumentere transaksjonen, samt muligheten til å legge inn tags som skal brukes i analyse. Vi på eksisterende løsninger og guider på nett for å lage den første siden. Metoden vi brukte til å konvertere transaksjonsloggen til Nordnet fungerer bra. Denne løsningen forutsetter at Nordnet filen kommer i norsk format, og prosesseres til et engelsk format. Det er viktig å være oppmerksom på dette ved installasjon på serveren.

For å vise transaksjonsloggen på siden kom vi fram til det beste ville være å lage en dynamisk tabell i JavaScript istedenfor å lage en statisk liste i HTML. Dette ble gjort som følge av at lengden på en transaksjonslogg vil variere mellom brukere. Ettersom det ikke er kodet inn noen begrensninger på transaksjonsloggen så kan dette medfører at loggen kan bli stor og uoversiktlig med dårlig responstid. En mulig løsning på dette er å innføre en begrensning på for eksempel 30 transaksjoner fordelt over flere faner. Denne løsningen kan også dele opp transaksjonsloggen slik at den er oppdelt i lister på 30. Med denne metoden vil første fane være klar mens de resterende fanene blir forberedt i bakgrunn. Det burde også være en mulighet å slette transaksjoner.Ved å implementere disse løsningene kan brukeren få en bedre oversikt over transaksjonene og bedret responstiden ved store transaksjonslogger.

For å gi brukeren muligheten til å velge tags på transaksjonene så vi på forskjellige løsninger. Vi valgte JavaScript pluginen Chosen, som gir flere muligheter til å lage dropdown menyer. I disse menyene kan man velge flere tags før man sender til serveren. Lagring av personlige kommentarer ble gjort ved å lage et tekstfelt. Ved videre utvikling kan det vurderes å endre tags til brukerdefinerte nøkkelord, da de nåværende tags'ene er forhåndsdefinerte.

5.1.1.3 Logg analyse

Når transaksjonsloggen blir stor og strekker seg over en lang periode, kan grafene bli smale og vanskelig å lese. Det kan løses ved å legge inn brukerdefinerte tidsrom for datauttrekket. Brukeren kan da få en mer informativ analyse.

5.1.2 Backend

Dette kapittelet beskriver prosesser, utfordringer og løsninger som vi stod ovenfor da vi utviklet elementer i backend. Samtidig som det diskuteres mulig forbedringspotensialer.

5.1.2.1 Web Server

Implementasjonen av webserveren fungerte ganske bra, uoptimalisert kode førte til litt treghet, men den leverer det som forventes av den og er skrevet på en forståelig og oversiktlig måte. Vi tok i bruk microservices basert på tidligere erfaring av samme type system. Microservices gjør det hele til en veldig oversiktlig og ukomplisert prosess å legge til mer

funksjonalitet, og å redigere allerede eksisterende funksjonalitet. Den største utfordringen vi møtte med webserveren var i begynnelsen da vi fikk problemer med CORS error. CORS (Cross-Origin Resource Sharing) er en mekanisme som webserveren bruker for å bestemme hvilke kilder andre enn seg selv den skal godta forespørsler fra. CORS error oppstår når en kilde som ikke er den selv sender forespørsler uten header parameter som spesifiserer at den har rettighet til det. Dette løste vi med reverse proxy ved å plassere websiden på et NGINX docker image og webserveren på et Payara docker image i stedet for å direkte drifte de fra en Ubuntu maskin.

Videre finnes det forbedringspotensialer på webserveren. Queries til databasen skulle vært utnyttet på en bedre måte. I noen tilfeller sorteres data direkte i forespørsel-metoden, det kunne blitt effektivisert ved å skrive å inkludere sortering i querien som ville gjort den samme jobben. I tillegg sendes ofte hele listen av innhold ved forespørsler, det er greit dersom det er lite informasjon. Men i tilfeller med mye informasjon påfører dette lang responstid, og det er synlig spesielt for aksjeloggen. Ved begrensing av antall entiteter som skal hentes ut, og sorteringsmetoder direkte i querien vil responstiden forbedres. Det vil også bidra til mindre, og mer oversiktlig kode, da det ofte er enklere å forstå en query enn en lang kode som gjør det samme. I loggen finnes det elementer en ikke vil ha med i oversikten sin (renteutbetalinger, overføringer m.m), disse er vanskelig å filtrere ut ved å skrive queries. Derfor er det alltid nødvendig å også skrive kode for å gjøre denne jobben.

Autorisasjonssystemet er designet for å ta i bruk brukergrupper, vi ble nødt til å arbeide med andre prioriterte oppgaver fremfor å fullføre denne jobben. Vi valgte likevel å la den koden stå, og gi alle brukerne i systemet en standard rolle. Videreutvikling av systemet vil få bruk for roller.

5.1.2.2 Database

Det oppstod problemer i begynnelsen da vi skulle implementere databasen til webserveren, lenge slet vi med å få mySQL database til å fungere på Payara serveren.Problemet vi møtte på var konfigurering av “connection pools” som skal sørge for at kobling mellom database og server holdes i live i stedet for å kaste bort masse ressurser på å opprette koblinger hver gang informasjon skal aksesseres (Progress, u.d.). Etter mye feilsøking og tid brukt på å løse problemet der konfigurasjonen av connection pools ikke fikk kontakt med databasen, gikk vi over til H2 da vi følte det var viktigere å prioritere andre oppgaver. H2 er en databaseløsning som Payara kommer med som standard. Denne databasen har noen ulemper, hovedsakelig at

den blir satt opp som en “in memory database” som betyr at den ikke lagres på disk og dermed best egnet til å bli brukt i utviklingsfasen (tutorialspoint, u.d.). Målet var å komme tilbake til dette, men vi tok en beslutning om å prioritere andre oppgaver siden vi allerede hadde en database som fungerte. For å kunne skalere prosjektet og optimalisere det for faktisk bruk, er det etterhvert nødvendig å bytte ut databasesystemet.

5.1.2.3 Nettskraper

Ingen på gruppen hadde kunnskap om nettskraping og hvordan dette er bygget opp. Av den grunn så krevde det flere iterasjoner med undersøkelser og koding av nettskraperen før gruppen ble fornøyd. De første iterasjonene var relativt enkle, ettersom de bare søkte på aksjer og hentet ut lenker, datoer og titler. Ved hver iterasjon lærte gruppen mer og mer slik at koden ble mer objekt-orientert, og funksjonaliteten ble forbedret. Den siste iterasjonen av nettskraperen hadde implementert all ønsket funksjonalitet. Flere nettaviser ble forespurt av veileder, ettersom gruppen ønsket mer tyngde bak forespørselen. Kun Dagens Næringsliv svarte, og gav tillatelse da vi spurte om lov til å skrape nettsiden deres. De gav også prosjektet en pluss-konto, slik at vi fikk tilgang til innholdet i alle artiklene deres. Det var ikke i henhold til planen å skrape kun en nettavis, da dette kan gi en ensidig vinkling av aksjemarkedet. Små aksjer kan få liten omtale i spesifikke medier og derfor vil disse aksjene kunne gi få eller ingen treff.

Nettskraperen kunne teknisk sett skrapet andre nettsider som E24 og Finansavisen, men som gruppe ville vi ikke bruke ressursene til nettaviser uten deres godkjenning. Likevel er den designet på en slik måte at man enkelt kan skrape andre nettsteder ved å endre/legge til hvilke XPaths som blir lagret i utgiver-objektet. Ved en framtidig videreutvikling ville vi jobbet hardere for å få godkjenning til å skrape andre kilder.

I ettertid ble det også oppdaget at de forskjellige artiklene som legges ut på Dagens Næringsliv har forskjellig struktur basert på om de er en kommentar fra redaktør eller en ren artikkel. Ettersom dette ble oppdaget sent i prosjektet, blir ikke innholdet fra kommentarer hentet ut og sendt til sentiment vurdering. Som konsekvens vil ikke denne artikkelen ikke ha en sentiment vurdering i systemet. I en framtidig løsning ville vi kodet inn en test for hvilken struktur artikkelen har og implementert flere XPaths i utgiver-objektet.

For å kunne tilby enda mer oppdatert informasjon til brukeren, kan man i en fremtidig versjon vurdere å abonnere på nettsider som utgir pressemeldinger tidlig. Slik kan man presentere artikler før de ordinære nettavisene omtaler disse nyhetene.

5.1.2.4 Artikkel sentiment

Vi har implementert muligheten til å bygge inn sentiment i systemet, og har lagt inn en grunnleggende analyse som "Proof of Concept". Årsaken til at en fungerende sentimentanalyse ikke ble implementert er at andre funksjoner måtte prioriteres. Fordelen med en slik analyse er at man som bruker kan få inntrykk av artikkelens innhold, og om dette lover positivt eller negativt for aksjen.

Vi har valgt en forenklet løsning der alle ord er vektet likt. Dette fungerer godt nok som en "Proof of Concept", men ikke godt nok til å stole på resultatene. For eksempel blir ordet "konkurs" og "svak" vektet likt mens i praksis vil "konkurs" være meget negativ og"svak" kan brukes i en positiv sammenheng. I en fremtidig versjon så kunne man løst dette ved å vekte ord avhengig av hvert enkelt ords betydning. En annen løsning kunne vært å laget et system som analyserer den underliggende meningen i en artikkel ved hjelp av beregnings lingvistikk. Under undersøkelsene vi gjorde i starten av prosjektet fant vi en Nevro Lingvistisk Programmering (NLP) ressurs, Norwegian NLP Resources, på GitHub.

5.2 Utviklingsprosess

Med unntak av NGINX var alle aktiviteter identifisert i forprosjektrapporten, men estimatene var ikke realistisk da gruppen ikke hadde nok erfaring innenfor denne type prosjekt til å estimere riktig. I tillegg er det ikke estimert en usikkerhet på estimatene. Det totale estimatet i forprosjektrapporten ligger på 833 timer, mens den realistiske tidsbruken er tilnærmet dobbelt så høy. I et framtidig prosjekt ville vi involvert en person med relevant erfaring under estimering, samtidig ville vi gjennomført en nøyere timeoppfølging, slik at man kan lære hvor lang tid de forskjellige oppgavene faktisk tar.

Vi brukte en Discord server til å kommunisere med hverandre. Det var her gruppemedlemmene kommuniserte på en daglig basis, men de daglige møtene ble ikke holdt på typiske 15 minutter. Istedenfor var gruppemedlemmene ofte aktive på denne serveren hele dagen og man ble med og forlot disse samtalene når man ville. Ved å ha en så tett kommunikasjon visste alle medlemmene som regel hva de andre jobbet med til enhver tid. Likevel ville det vært bedre for utviklingsprosessen om vi hadde vært samlet i det samme lokalet, siden det ville vært enklere å hjelpe hverandre og samkjørt visjonen om hvordan sluttproduktet skulle blitt.

Scrum-board ble brukt til planlegging av de forskjellige oppgavene som skulle utføres, og til planlegging av hvilke oppgaver som skulle inn i hvilke sprinter. Fra starten av prosjektet

hadde gruppen neglisjert å velge en Scrum-master. Dette førte til at ingen fulgte med på om sprintene var startet eller avsluttet i Jira. I tillegg var det et gjentagende problem at gruppemedlemmene ikke flyttet på jobbene i Jira underveis i en sprint. Dette førte til sprinter ble startet og avsluttet til feil tid i verktøyet, samt at stories og tasks ikke ble flyttet inn "under arbeid" eller "ferdige" før sprint ble avsluttet. Konsekvensen av dette var at man ikke var sikre på hva de andre gruppemedlemmene jobbet med om de ikke dukket opp i møtene en dag. I verste fall hadde dette kunne ført til at flere hadde jobbet på samme oppgave, men dette skjedde ikke i praksis. Erfaringen fra dette er at en Scrum-master er nødvendig for optimal flyt i utviklingen, samt at alle medlemmene i teamet oppdaterer status på sine oppgaver.

Sprint review ble gjennomført hver 14 dag og ble brukt til å demonstrere funksjonaliteten som ble implementert i løpet av sprintene for oppdragsgiver og veileder. Under disse møtene ble neste sprint planlagt og oppdragsgiverens ønsker ble lagt vekt på ved valg av stories og funksjonalitet som skulle implementeres. Gruppen erfarte å få gode tilbakemeldinger på hva som var bra og hva som måtte forbedres på disse møtene slik at sluttproduktet ble bedre. Arbeidsmengdene per sprint var godt tilpasset tilgjengelig tid og prosjektgruppen.

Enkelte oppgaver tok ekstra lang tid, spesielt analyse av transaksjonsloggen. Denne oppgaven ble først utført av gruppen, men denne løsningen viste seg å ikke være god nok til å opprettholde kravene. Dette førte til at det måtte avsettes tid fra neste sprint til å lage denne analysen på nytt fra eksempler som ble tilsendt gruppen fra veileder og oppdragsgiver. Tiden vi mistet på grunn av dette hadde vi kunne utnyttet mot slutten av prosjektet til å utbedre potensielle feil og mangler.

6 KONKLUSJON

De valgte teknologiene var egnet til å løse de ulike funksjonelle kravene. Nettskraperen fungerer bra, men samler fra få kilder grunnet manglende godkjenning fra redaktørene hos nettavisene. Selve Scrum-prosessen har fungert godt, til tross for at rollen Scrum-master bare har vært delvis fylt. Grunnet Korona-pandemien har gruppen kun jobbet distribuert, noe som kan ha påvirket sluttproduktet. Prosjektet bommet på estimert tid og timeoppfølging sviktet, dette påvirket viktig læring opp mot neste prosjekt.

Systemet har løst problemstillingen, og har nok funksjonell dekning til å kunne tas i bruk.

7 REFERANSER

Apache Maven, 2021. *Apache Maven Project.* [Internett]
Available at: https://maven.apache.org/maven-features.html
[Funnet 11 5 2021].

babu, J., 2019. *How to convert Excel file data into a JSON object by using JavaScript.* [Internett]
Available at: https://levelup.gitconnected.com/how-to-convert-excel-file-into-json-object-by-using-javascript-9e95532d47c5
[Funnet 20 Mai 2021].

Baeldung, 2020. *Introduction to Project Lombok.* [Internett]
Available at: https://www.baeldung.com/intro-to-project-lombok
[Funnet 11 5 2021].

BEKK Open Radar, 2018. *Pull-request.* [Internett]
Available at: https://radar.bekk.no/tech2016/prosess-og-kvalitet/pull-requests
[Funnet 20 Mai 2021].

Cawley, C., 2018. *Windows 10 SSH vs. PuTTY: Time to Switch Your Remote Access Client?.* [Internett]
Available at: https://www.makeuseof.com/tag/windows-10-ssh-vs-putty/
[Funnet 20 Mai 2021].

Christensson, P., 2012. *Java.* [Internett]
Available at: https://techterms.com/definition/java
[Funnet 03 Mai 2021].

Christensson, P., 2015. *HTML.* [Internett]
Available at: https://techterms.com/definition/html
[Funnet 05 Mai 2021].

Cloudflare, u.d. *What Is A Reverse Proxy? | Proxy Servers Explained.* [Internett]
Available at: https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/
[Funnet 8 11 2021].

CLOUDNWEB, u.d. *Nginx for Front-end Developers - Beyond the Basics.* [Internett]
Available at: https://cloudnweb.dev/2019/06/nginx-for-front-end-developers-beyond-the-

basics/
[Funnet 8 5 2021].

Dash Enterprise, 2021. *Plotly JavaScript Open Source Graphing Library.* [Internett]
Available at: https://plotly.com/javascript/
[Funnet 05 Mai 2021].

Eclipse Foundation, u.d. *Eclipse Jersey.* [Internett]
Available at: https://eclipse-ee4j.github.io/jersey/
[Funnet 11 5 2021].

Enginess, 2014. *The 6 Principles Of Design, a la Donald Norman.* [Internett]
Available at: https://www.enginess.io/insights/6-principles-design-la-donald-norman
[Funnet 20 Mai 2021].

Gilstrap, B., 2009. *AN INTRODUCTION TO JAX-RS AND JERSEY.* [Internett]
Available at: https://objectcomputing.com/resources/publications/sett/august-2009-an-introduction-to-jax-rs-and-jersey
[Funnet 11 05 2021].

GURU99, u.d. *RESTful Web Services Tutorial with REST API Example.* [Internett]
Available at: https://www.guru99.com/restful-web-services.html
[Funnet 9 5 2021].

Guru99, u.d. *What is DBMS? Application,Types,Example,Advantages.* [Internett]
Available at: https://www.guru99.com/what-is-dbms.html
[Funnet 20 Mai 2021].

Hartwig, K., 2021. *Nye kunder i Nordnet doblet i første kvartal: – Opplever en voldsom interesse for sparing og investering.* [Online]
Available at: https://www.dn.no/marked/sparing-og-investering/dn-aksjer/bors/nye-kunder-i-nordnet-doblet-i-forste-kvartal-opplever-en-voldsom-interesse-for-sparing-og-investering/2-1-1002847
[Accessed 20 Mai 2021].

Harvest, 2016. *Chosen.* [Internett]
Available at: https://harvesthq.github.io/chosen/
[Funnet 05 Mai 2021].

Hjertaker, O., 2018. *afinn.* [Internett]
Available at: https://github.com/olavski/afinn/blob/master/afinn/data/AFINN-no-165.txt
[Funnet 20 Mai 2021].

Itgoslo, 2019. *norsentlex.* [Internett]
Available at: https://github.com/ltgoslo/norsentlex/tree/master/Fullform
[Funnet 20 Mai 2021].

Jackson, B., 2017. *Web Crawlers and User Agents - Top 10 Most Popular.* [Internett]
Available at: https://www.keycdn.com/blog/web-crawlers
[Funnet 03 Mai 2021].

JetBrains, u.d. *Essential tools for software developers and teams.* [Internett]
Available at: https://www.jetbrains.com/
[Funnet 20 Mai 2021].

JWT, u.d. *Introduction to JSON Web Tokens.* [Internett]
Available at: https://jwt.io/introduction
[Funnet 20 Mai 2021].

Karankc27, 2021. *How to Convert CSV to JSON file having Comma Separated values in Node.js ?.* [Online]
Available at: https://www.geeksforgeeks.org/how-to-convert-csv-to-json-file-having-comma-separated-values-in-node-js/
[Accessed 20 Mai 2021].

Lamouchi, N., 2016. *NetBeans 8.1 versus IntelliJ 15.* [Internett]
Available at: https://jaxenter.com/netbeans-8-1-versus-intellij-15-123263.html
[Funnet 8 5 2021].

Millington, S., 2021. *A Solid Guide to SOLID Principles.* [Internett]
Available at: https://www.baeldung.com/solid-principles
[Funnet 20 Mai 2021].

Monteiro, J.-L., 2018. *What is Eclipse MicroProfile?.* [Internett]
Available at: https://www.tomitribe.com/blog/what-is-eclipse-microprofile/
[Funnet 11 5 2021].

NGINX, u.d. *What is NGINX?*. [Internett]
Available at: https://www.nginx.com/resources/glossary/nginx/
[Funnet 8 5 2021].

octoparse, 2020. *octopase.* [Online]
Available at: https://www.octoparse.com/
[Accessed 19 05 2021].

parsehub, n.d. *parsehub.* [Online]
Available at: https://www.parsehub.com/
[Accessed 16 5 2021].

Paul, J., 2020. *10 Coding Principles Every Programmer Should Learn.* [Internett]
Available at: https://dzone.com/articles/10-coding-principles-every-programmer-should-learn
[Funnet 20 Mai 2021].

Perrin, C., 2008. *Use PuTTY as an SSH client on Windows.* [Internett]
Available at: https://www.techrepublic.com/blog/it-security/use-putty-as-an-ssh-client-on-windows/
[Funnet 5 5 2021].

Progress, u.d. *JDBC Connection Pooling.* [Internett]
Available at: https://www.progress.com/tutorials/jdbc/jdbc-jdbc-connection-pooling
[Funnet 20 Mai 2021].

Python Software Foundation, 2021. *What is Python? Executive Summary*. [Internett]
Available at: https://www.python.org/doc/essays/blurb/
[Funnet 05 Mai 2021].

RapidAPI, 2021. *How To Use the Yahoo Finance API in 2021*. [Internett]
Available at: https://rapidapi.com/blog/how-to-use-the-yahoo-finance-api/
[Funnet 20 Mai 2021].

Rapidapi, u.d. *Path Parameters – What are Path Parameters?*. [Internett]
Available at: https://rapidapi.com/blog/api-glossary/parameters/path/
[Funnet 10 5 2021].

Rapidapi, u.d. *Query Parameters – What are Query Parameters?.* [Internett]
Available at: https://rapidapi.com/blog/api-glossary/parameters/query/
[Funnet 10 5 2021].

repusate data in sight, 2021. *repusate.* [Online]
Available at: https://www.repustate.com/norwegian-sentiment-analysis/
[Accessed 15 5 2021].

Repustate, n.d. *NORWEGIAN SENTIMENT ANALYSIS.* [Online]
Available at: https://www.repustate.com/norwegian-sentiment-analysis/
[Accessed 20 Mai 2021].

Restfulapi, u.d. *REST Resource Naming Guide.* [Internett]
Available at: https://restfulapi.net/resource-naming/
[Funnet 9 55 2021].

Robie, J., Dyck, M. & Spiegel, J., 2021. *XML Path Language (XPath) 3.1.* [Internett]
Available at: https://www.w3.org/TR/xpath-31/
[Funnet 07 Mai 2021].

Rossen, E. & Øverby, H., 2020. *RSS.* [Internett]
Available at: https://snl.no/RSS
[Funnet 07 Mai 2021].

scrapy, n.d. *scrapy.* [Online]
Available at: https://scrapy.org/
[Accessed 16 05 2021].

selenium, 2021. *slenium.* [Online]
Available at: https://www.selenium.dev/
[Accessed 18 05 2021].

stormcrawler, 2021. *stormcrawler.* [Online]
Available at: http://stormcrawler.net/
[Accessed 18 05 2021].

TechTarget Contributor, 2017. *Scrum.* [Internett]
Available at: https://searchsoftwarequality.techtarget.com/definition/Scrum
[Funnet 29 April 2021].

Thomas, M., 2019. *HOW AI TRADING TECHNOLOGY IS MAKING STOCK MARKET INVESTORS SMARTER.* [Internett]
Available at: https://builtin.com/artificial-intelligence/ai-trading-stock-market-tech
[Funnet 20 Mai 2021].

Turizo, F., 2019. *Microservices for Java EE Developers.* [Internett]
Available at: https://blog.payara.fish/microservices-for-java-ee-developers2
[Funnet 11 5 2021].

tutorialspoint, u.d. *H2 Database - Introduction.* [Internett]
Available at: https://www.tutorialspoint.com/h2_database/h2_database_introduction.htm
[Funnet 20 Mai 2021].

Visual Studio, u.d. *Visual Studio Documentation.* [Internett]
Available at: https://code.visualstudio.com/docs
[Funnet 8 5 2021].

W3Schools, 2021. *jQuery Introduction.* [Internett]
Available at: https://www.w3schools.com/jquery/jquery_intro.asp
[Funnet 05 mai 2021].

Wallen, J., 2020. *Ubuntu Server: A cheat sheet.* [Internett]
Available at: https://www.techrepublic.com/article/ubuntu-server-the-smart-persons-guide/
[Funnet 9 5 2021].

Web64, 2019. *NoReC - FastText Model.* [Internett]
Available at: https://github.com/web64/norec-fasttext
[Funnet 20 Mai 2021].

Wikipedia, 2021. *Jakarta RESTful Web Services.* [Internett]
Available at: https://en.wikipedia.org/wiki/Jakarta_RESTful_Web_Services
[Funnet 11 5 2021].

Wikipedia, 2021. *Stock exchange.* [Internett]
Available at: https://en.wikipedia.org/wiki/Stock_exchange
[Funnet 28 April 2021].

Wikipedia, 2021. *Web scraping.* [Internett]
Available at: https://en.wikipedia.org/wiki/Web_scraping
[Funnet 03 Mai 2021].

Williams, B., 2016. *Intro to Databases (for people who don't know a whole lot about them).* [Internett]
Available at: https://medium.com/@rwilliams_bv/intro-to-databases-for-people-who-dont-know-a-whole-lot-about-them-a64ae9af712
[Funnet 10 5 2021].

VEDLEGG

Vedlegg 1 Kildekode Stock-trader-webpage.zip

Vedlegg 2 Forprosjektrapport

Vedlegg 3 Kravspesifikasjon

Vedlegg 4 Sprint rapporter

Vedlegg 5 ER-modell