

ITPE-3200 Høsten 2022

Webapplikasjoner

Oppretting av en aksjehandelswebapplikasjon (MVP).

Kandidater: 192, 178, 176, 195, 280.

ITPE-3200 Høsten 2022	1
Valg av oppgave og kravspesifikasjoner:	3
Tidlig fase:	3
UML	3
Prototype av nettsiden med Figma	4
Samarbeid og fordeling av oppgaver	5
Frontend:	5
Oversikt over alle sider	6
Javascript:	15
Backend:	18
Databasestruktur	18
Hva er CRUD og hvordan har vi implementert CRUD?	23
Hvordan er systemet bygget opp?	23
DBInit	32
Tilleggsfunksjonalitet	32
Konklusjon:	33
Kilder:	33

Valg av oppgave og kravspesifikasjoner:

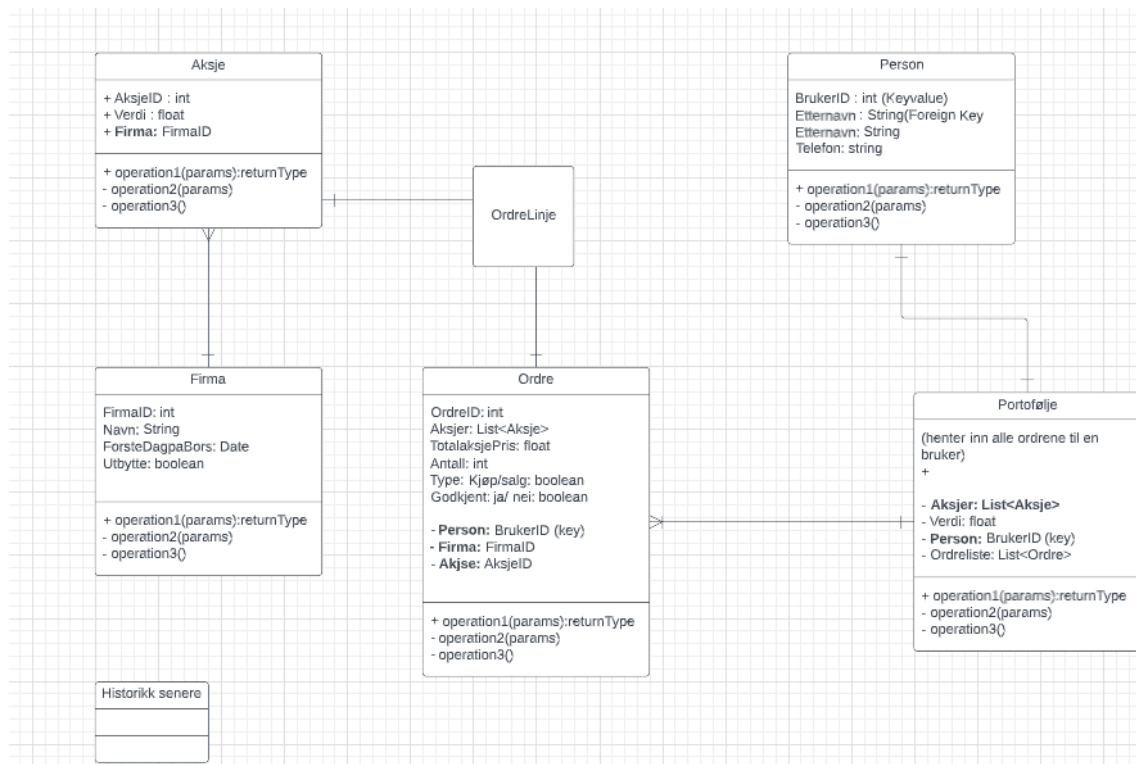
Gruppen har valgt å utvikle en aksjehandelswebapplikasjon med følgende brukerkrav:

- En bruker skal kunne logge seg inn med e-postadressen.
- En bruker skal kunne opprette en profil.
- En bruker skal kunne logge seg inn med en opprettet profil.
- En bruker skal kunne se hvilke aksjer som er til salgs.
- En bruker skal kunne få en oversikt over sine egne aksjer.
- En bruker skal kunne kjøpe en eller flere aksjer.
- En bruker skal kunne selge en eller flere aksjer.
- En bruker skal kunne endre opplysninger om seg selv på sin egen profil.
- En bruker skal kunne slette sin egen profil.

Tidlig fase:

UML

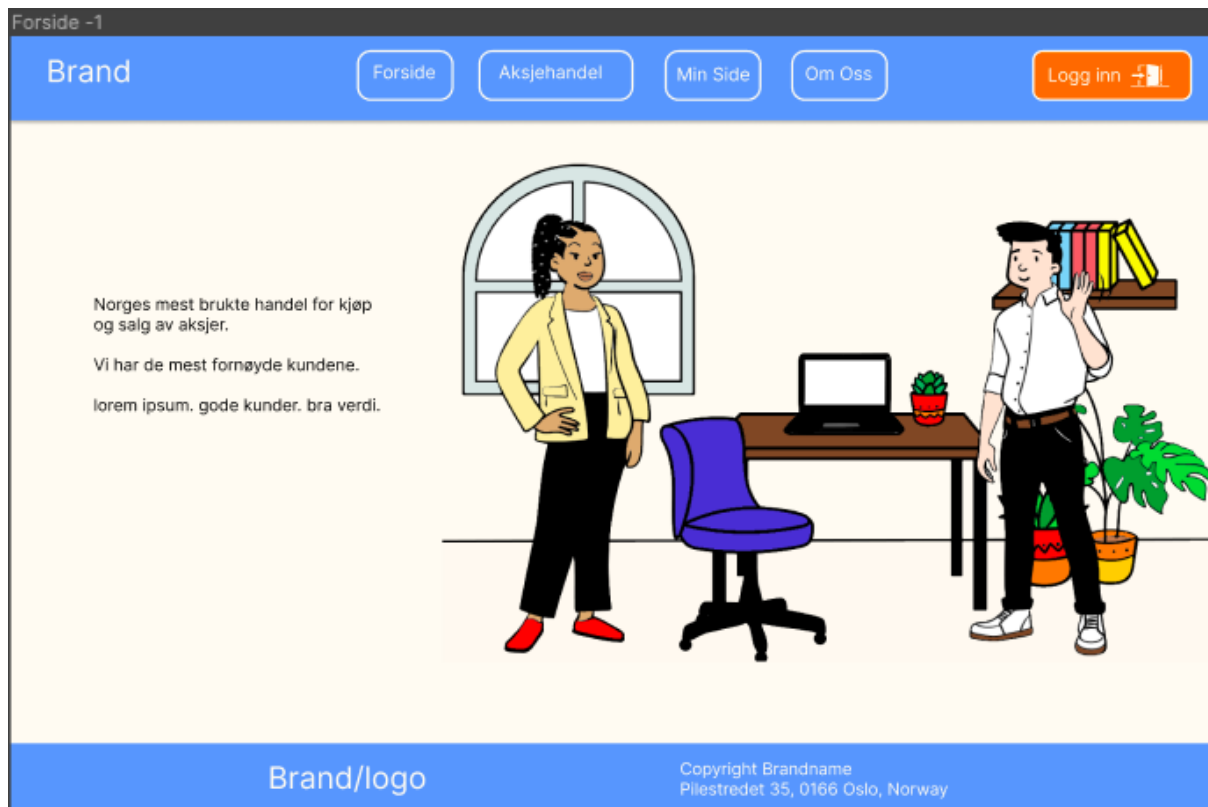
Vi startet med å strukturere og visualisere hvordan relasjonsdatabasen skulle se ut, ved hjelp av klassediagrammer. Dette gjennomførte vi ved hjelp av programmet LucidCharts. I løpet av utviklingsprosessen ble dette diagrammet senere endret på.



Første utkast av relasjonsdatabasen.

Prototype av nettsiden med Figma

Videre begynte vi med å sette opp et utkast av nettsiden i designprogrammet Figma. De grafiske figurene på nettsiden har vi laget selv ved å sette sammen ikoner som er tilgjengelige for bruk i Microsoft.



Utkast av nettsiden i Figma

Samarbeid og fordeling av oppgaver

Videre satt vi opp et GitHub-prosjekt, for å gjøre det enkelt å samarbeide med utviklingen av prosjektet. Vi delte oss opp i et front-end og et back-end team. Teamene jobbet tett sammen gjennom hele prosessen, med god kommunikasjon, for at sluttproduktet skulle stå til forventningene.

Frontend:

På frontend-delen av prosjektet har vi brukt generell HTML, JavaScript og CSS. Vi har også tatt i bruk rammeverket Bootstrap, som forenkler prosessen for å lage en responsiv side. Nettsiden har gått gjennom flere faser hvor vi har videreutviklet funksjoner og revidert tidligere versjoner.

Vi bestemte oss underveis for å begrense funksjonalitet om man ikke er logget inn med en registrert bruker. Vi har lagt til en bruker i databasen (logg inn ved 123@gmail.com), men har også lagt til muligheten til å registrere en ny bruker. Som innlogget vil man få tilgang til egen profil, få en oversikt over aksjer, og mulighet for salg eller kjøp av disse, og tilgang til egen portefølje. Vi har ikke implementert sikkerhet rundt brukere og innlogging, men knytter opp handelshistorikk, portefølje og andre tilhørende dataer til en bruker-ID.



Hjem-siden uten å være logget inn.



Hjem-siden hos en innlogget bruker.

Oversikt over alle sider

I tabellen under vises alle HTML-sidene, hva de gjør, og deres tilhørende JavaScript-filer som blir brukt i den endelige versjonen av webapplikasjonen.

Uten å være innlogget:

index.HTML	Applikasjonens hjemmeside, hvor man finner informasjon om hvilken nettside man er på, med mulighet til å navigere til de andre fanene.	JavaScript funksjoner + koblinger
omOss.HTML	Statisk informasjon om bedriften og kontaktinformasjon.	
LoggInn.HTML	Brukeren kan logge inn, eller trykke seg videre til registrering.	loggInn.js <ul style="list-style-type: none">- validerEmail()- loggInn()
registrerBruker.HTML	En side for å registrere ny bruker.	registrerBruker.js <ul style="list-style-type: none">- valiserPerson()- lagrePerson()- leggTilPerson()

Innlogget:

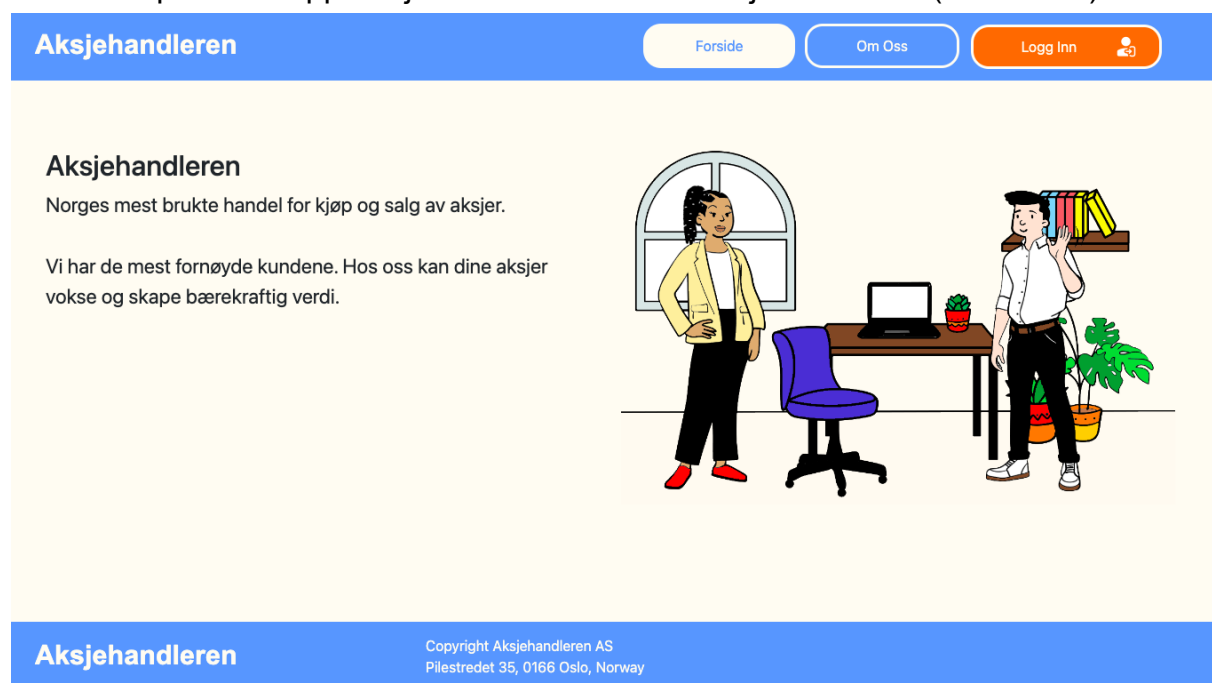
indexLoggetInn.HTML	Webapplikasjonens hjemmeside som innlogget. Man har da flere valgmuligheter i navigasjonsbaren.	
omOssLoggetInn.HTML	Webapplikasjonens Om oss-side som innlogget. Man har da flere valgmuligheter i navigasjonsbaren.	
aksjehandel.HTML	Siden gir en oversikt over aksjetabellen, med aksjenavn og verdi, i tillegg til mulighet for å kjøpe og selge.	Aksjehandel.js <ul style="list-style-type: none">- hentAlleAksjene()- validerTall()- kjøp()- selg()
visPortefolje.HTML	Siden gir en oversikt over	visPortefolje.js

	brukerens aksjeportefølje, i tillegg til en dynamisk avkastning som oppdateres gjenvlig.	<ul style="list-style-type: none"> - hentAlleAksjene() - formaterAksjer()
visOrdre.HTML	Siden gir en oversikt over brukerens handelshistorikk.	visOrdre.js <ul style="list-style-type: none"> - hentAlleOrdrene() - formaterOrdrene()
registrerEndringer.HTML	En bruker kan redigere brukerinformasjon og lagre nye oppdateringer.	registrerEndringer.js <ul style="list-style-type: none"> - validerPerson() - hentEnPerson() - formaterPerson() - rediger()
bestilling.HTML	Når en bruker legger inn et kjøp på aksjehandel.HTML videreføres man til en bekreftelsesside med ordreinformasjon og mulighet til å bekrefte eller avkrefte kjøpet.	kjopBekreftelse.js <ul style="list-style-type: none"> - hent() - formatter() - ikkeGodkjenn() - bekreftOrdre()
bestillingSalg.HTML	Når en bruker legger inn et salg på aksjehandel.HTML videreføres man til en bekreftelsesside med salgsinformasjon og mulighet til å bekrefte eller avkrefte salget.	salgBekreftelse.js <ul style="list-style-type: none"> - hent() - formatter() - ikkeGodkjenn() - bekreftOrdre() - registrerOrdre() - forLite() - harIkke()
minSide.HTML	Siden viser en oversikt over brukerinformasjon, med mulighet til å redigere informasjonen eller slette brukeren. Det er i tillegg mulig å trykke seg videre inn på brukerens handelshistorikk, eller å logge ut.	minSide.js <ul style="list-style-type: none"> - hentEnPerson() - formatterPerson() - slettBruker() - loggUt() - seOrdre()

Vi har brukt CSS til å designe sidene. Alle sidene er koblet til index.css som inneholder alt av generelt design; headere, footere, body-bakgrunner og navigasjonsmeny. Sidene med større spesifiseringsbehov fikk i tillegg tilhørende .css-filer.

Applikasjonens gang

Når man åpner webapplikasjonen møtes man med hjemmesiden (index.html).



Videre kan man benytte navigasjonsbaren for å trykke seg videre til *Om oss*-siden (omOss.html).

Aksjehandleren

[Forside](#)[Om Oss](#)[Logg Inn](#)

Om Oss

Bedriften ble startet i 2020. Vi holder til i nyoppussede lokaler ved Pilestredet 35 i Oslo.
Kontakt oss på telefon: 40405050.
Send oss en e-post på: kontakt@aksjehandleren.no



Aksjehandleren

Copyright Aksjehandleren AS
Pilestredet 35, 0166 Oslo, Norway

Man kan også trykke seg inn på *Logg inn*-siden ([loggInn.html](#)).

Aksjehandleren


[Forside](#)[Om Oss](#)[Logg Inn](#)

Logg inn her:

E-post:

[Logg inn](#)

Ny bruker? Registrer deg her:

[Registrer deg her](#)


Aksjehandleren

Copyright Aksjehandleren AS
Pilestredet 35, 0166 Oslo, Norway

Her kan man logge inn med eksisterende bruker eller registrere seg ved å trykke på registrer-knappen. Den tar deg videre til en registreringsside ([registrerBruker.html](#)).

Aksjehandleren

Fornavn:

Ole

Etternavn:

Tarje

Telefon:

44334433

Registrer endringer

Tilbake

Aksjehandleren

Copyright Aksjehandleren AS
Pilestredet 35, 0166 Oslo, Norway

Man registrerer seg ved å legge inn brukerinformasjon som kreves.

Etter at man har registrert en bruker er det mulig å logge seg inn. Hjemmesiden som innlogget har flere valgmuligheter i navigasjonsbaren (indexLoggetInn.html).

Aksjehandleren

Forside

Aksjehandel

Portefølje

Om oss

Min side

Aksjehandleren

Norges mest brukte handel for kjøp og salg av aksjer.

Vi har de mest fornøyde kundene. Hos oss kan dine aksjer vokse og skape bærekraftig verdi.



Aksjehandleren

Copyright Aksjehandleren AS
Pilestredet 35, 0166 Oslo, Norway

Man får blant annet mulighet til å trykke seg inn på siden for aksjehandel (aksjehandel.html).

Aksjehandleren				
<div>ForsideAksjehandelPorteføljeOm ossMin side</div>				
Aksjehandel:				
Aksje	Pris	Antall	Kjøp	Salg
Equinor	6.91	<input type="text"/>	Kjøp	Salg
Tesla	12.46	<input type="text"/>	Kjøp	Salg
Apple	14.92	<input type="text"/>	Kjøp	Salg
Kahoot	18.19	<input type="text"/>	Kjøp	Salg
Samsung	9.06	<input type="text"/>	Kjøp	Salg
Amazon	14.94	<input type="text"/>	Kjøp	Salg
Google	19.74	<input type="text"/>	Kjøp	Salg


Her kan man selge og kjøpe aksjer. Kjøp- og salgsknappene viderefører brukeren til en bekreftelsesside.

Her vises bestillingen, hvor man kan godkjenne kjøpet eller salget. Etter å ha bekreftet eller avkreftet bestillingen tas brukeren tilbake til aksjehandelssiden.

Aksjehandleren	
<div>Tilbake</div>	<div><div>Din bestilling:</div><div>Kjøp: 1 Equinor aksjer (6.8 NOK per aksje)</div><div>Totalpris: 6.80 kr.</div><div><div>Godkjenn</div><div>Ikke godkjenn</div></div></div>
<div>Aksjehandleren</div> <div>Copyright Aksjehandleren AS Pilestredet 35, 0166 Oslo, Norway</div>	

Portefølje-knappen i navigasjonsbaren fører brukeren videre til porteføljesiden med oversikt over hvilke aksjer man eier, totalverdi og avkastning (visPortefolje.html).

Aksjehandleren

[Forside](#)[Aksjehandel](#)[Portefølje](#)[Om oss](#)[Min side](#)

Dine aksjer:

Aksje	Antall	Verdi
Equinor	5	36.35
Tesla	10	119.3
Verdi:	15	155.65

Din avkastning:

Type	Antall	Verdi
Kjøpssum:	15	155.50
Avkastning:	0.15 NOK	0.10%

Videre kan brukeren trykke seg inn på Min Side (minSide.html)

Aksjehandleren

[Forside](#)[Aksjehandel](#)[Portefølje](#)[Om oss](#)[Min side](#)

Profil

Ole

Tarje

44334433

oletarje@gmail.com

Rediger informasjon

Slett bruker

[Handelshistorikk](#)

[Logg ut](#)

Aksjehandleren

Copyright Aksjehandleren AS
Pilestredet 35, 0166 Oslo, Norway

Siden gir en oversikt over brukerinformasjon, med mulighet for å redigere opplysningene sine, slette brukeren sin, vise handelshistorikk eller logge ut. Ved å trykke på *Handelshistorikk* tas brukeren videre til visOrdre.html. Her har man en oversikt over all handelshistorikk tilknyttet brukeren.

Man har også muligheten til å endre brukeropplysninger ved å trykke på *Rediger informasjon*-knappen.

Aksjehandleren

Fornavn:

Ole

Etternavn:

Tarje

Telefon:

44334433

Registrer endringer

Tilbake

Ved å trykke på *Registrer endringer* vil disse bli lagret.

Aksjehandleren

[Forside](#)[Aksjehandel](#)[Portefølje](#)[Om oss](#)[Min side](#)

Tilbake

Dine kjøp:

Dato	Antall	Navn
0/9/2022 18:51:3	1	Equinor

Dine salg:

Dato	Antall	Navn
------	--------	------

Aksjehandleren

Copyright Aksjehandleren AS
Pilestredet 35, 0166 Oslo, Norway

Vi har også laget en navigasjonsbar for mindre skjermer.



Slik blir det enkelt og brukervennlig å navigere mellom sidene - også på mindre skjermer.

Javascript:

Funksjoner og hva de gjør:

logInn.js

- `validerEmail()`: tar inn en e-post og kontrollerer formatet ved hjelp av en `RegExp`. Returnerer `true` hvis formatet er riktig og `false` hvis den ikke er riktig.

- `loggInn()`: henter e-post og validerer den. Logger inn hvis validering viser seg å være gyldig, og gir en feilmelding om ikke.

`registrerBruker.js`

- `validerPerson()`: validerer formateringen gjennom if-tester, hvor brukeren får feilmelding om deres input-verdier ikke stemmer overens med RegEx.
- `lagrePerson()`: lagrer verdiene tilhørende en bruker, i tillegg til å validere brukeren. Man får en feilmelding ved feil formatering eller om brukeren allerede eksisterer.
- `leggTilPerson()`: sender et person-objekt til server-siden og lagrer brukeren i databasen. Får en feilmelding om det ikke er gjennomførbart.

`aksjehandel.js`

- `hentAlleAksjene()`: henter alle aksjene fra databasen, og kaller på `formaterAksjer()` for formatering.
- `oppdater()`: oppdaterer aksje-objektet, med priser og tilgjengelige aksjer.
- `validerTall()`: sjekker at antall aksjer er et gyldig tall ved bruk av RegEx.
- `kjop()`: lagrer en kjøpsbestilling og kaller på `validerTall()`.
- `selg()`: lagrer en salgsbestilling og kaller på `validerTall()`.
- `formaterAksjer()`: formaterer aksjetabellen og implementerer kjøp- og salgsknapper.
- `readText()`: lagrer verdiene i aksjetabellen ved hjelp av `sessionStorage` for å forhindre at verdiene forsvinner når tabellen oppdateres ved gjenvnlige intervaller.

`visPortefolje.js`

- `hentAlleAksjene()`: henter alle aksjene fra databasen, og kaller på `formaterAksjer()` for formatering.
- `formaterAksjer()`: formaterer aksjetabellen og implementerer kjøp- og salgsknapper.
- `hentAvkastning()`: formaterer avkastningen i en tabell tilsvarende porteføljen. Beregner totale avkastningen til porteføljen.

`visOrdre.js`

- `hentAlleOrdrene()`: henter ordrene fra `sessionStorage`. Kaller også på `formaterOrdrene()` for formatering.
- `formaterOrdrene()`: formaterer ordrene i to tabeller, en for kjøp og en for salg.

`registrerEndringer.js`

- `validerPerson()`: Kontrollerer at alle input-verdiene stemmer overens med tilhørende RegEx. Gir brukeren en feilmelding hvis en verdi er ugyldig.
- `hentEnPerson()`: henter en bruker fra `sessionStorage` og bruker-ID.
- `formaterPerson()`: formaterer brukeropplysninger.
- `rediger()`: henter brukeropplysningene tilhørende bruker-IDen og lager et nytt person-objekt. Objektet lagres deretter i databasen.

kjopBekreftelse.js

- hent(): henter aksjene ved aksje-ID og kaller på formatter() for formatering.
- formatter(): formaterer ordrebekreftelsen.
- ikkeGodkjenn(): tar brukeren tilbake til aksjehandelssiden.
- bekreftOrdre(): henter aksje-IDen, registrerer aksjen og antall i porteføljen og dato. Lagrer i databasen og returnerer brukeren tilbake til aksjehandelssiden.

salgBekreftelse.js

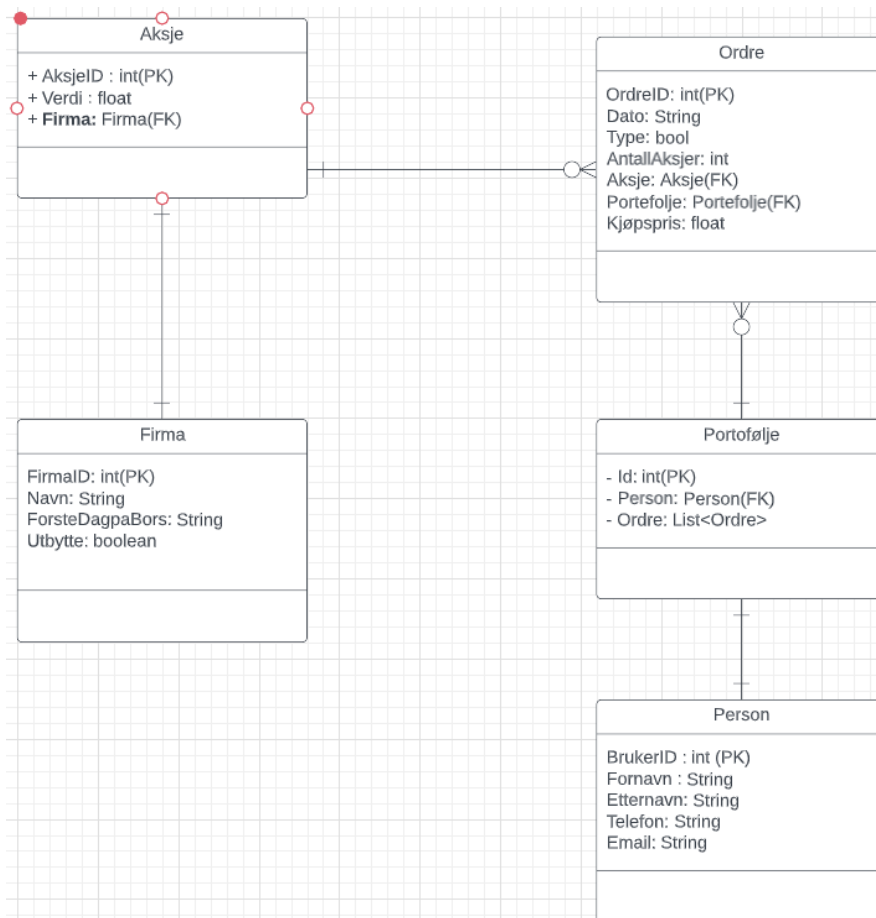
- hent(): henter aksjene ved aksje-ID og kaller på formatter() for formatering.
- formatter(): formaterer ordrebekreftelsen.
- ikkeGodkjenn(): tar brukeren tilbake til aksjehandelssiden.
- bekreftOrdre(): henter aksje-IDen, registrerer aksjen og antall i porteføljen og dato. Lagrer i databasen og returnerer brukeren tilbake til aksjehandelssiden. Kaller på forLite() og harIkke() for å kontrollere at en bruker ikke selger flere aksjer enn en allerede innehar. Lagrer ordren i databasen hvis mulig.
- registrerOrdre(): registrerer ordren hvis alle tester i bekreftOrdre() godkjennes.
- forLite(): sender feilmelding når en bruker prøver å selge flere aksjer enn de har selv.
- harIkke(): sender feilmelding når en bruker prøver å selge en aksje de ikke har.

minSide.js

- hentEnPerson(): henter brukeropplysningene til en bruker som er logget inn.
- formatterPerson(): henter brukeropplysninger via et kall til hentEnPerson(person) og gir mulighet til å revidere.
- slettBruker(): sletter en bruker og tilhørende brukerinformasjon og sletter den fra databasen.
- loggUt(): logger en bruker ut av applikasjonen, og sletter lagret ID fra sessionStorage.
- seOrdre(): tar brukeren til visOrdre.html.

Backend:

Vårt endelige klassediagram ble seende slik ut:




Det endelige Klassediagram

Databasestruktur

Vi har satt opp totalt fem tabeller i databasen - person.cs, portefolje.cs, ordre.cs, aksje.cs og firma.cs.

Aksje-tabell:

I aksje-tabellen lagres det en ID for hver aksje som primærnøkkel, med Firma-ID som fremmednøkkel og verdi.

Table:  Aksjer

	Id	Verdi	FirmaId
	Filter	Filter	Filter
1	1	7.0	1
2	2	12.0	2
3	3	15.0	3
4	4	18.0	4
5	5	9.0	5
6	6	15.0	6
7	7	20.0	7

```
Aksje.cs*  [icon] X
AksjeHandelWebApp  AksjeHandelWebApp.Models.Aksje
1  using System;
2  using System.Collections.Generic;
3
4  namespace AksjeHandelWebApp.Models
5  {
6      24 references
7      public class Aksje
8      {
9          4 references
10         public int Id { get; set; }
11         15 references
12         public float Verdi { get; set; }
13         9 references
14         public virtual Firma Firma { get; set; }
15     }
16 }
```

Firma-tabell:

I firma-tabellen lagres det også en ID som primærnøkkel, med navn på firmaet, når firmaet kom på børs og utbytte som andre verdier.

Table: Firmaer

	Id	Navn	ForsteDagPaBors	Utbytte
	Filter	Filter	Filter	Filter
1	1	Equinor	11.11.11	1
2	2	Tesla	11.11.11	1
3	3	Apple	11.11.11	1
4	4	Kahoot	11.11.11	1
5	5	Samsung	11.11.11	1
6	6	Amazon	11.11.11	1
7	7	Google	11.11.11	1

```

Firma.cs  x  Aksje.cs*
AksjeHandelWebApp  AksjeHandelWebApp.Models.Firma
1  using System;
2  namespace AksjeHandelWebApp.Models
3  {
4      public class Firma
5      {
6          public int Id { get; set; }
7          public string Navn { get; set; }
8          public string ForsteDagPaBors { get; set; }
9          public bool Utbytte { get; set; }
10     }
11 }

```

Ordre-tabell:

Ordre-tabellen består av en ID som primærnøkkel, med aksje-ID og portefølje-ID som fremmednøkler, med type, antall aksjer og kjøpspris som andre verdier.




```

Person.cs  x  Ordre.cs  Firma.cs  Aksje.cs*
AksjeHandelWebApp  AksjeHandelWebApp.Models.Person
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel.DataAnnotations.Schema;
4
5  namespace AksjeHandelWebApp.Models
6  {
7      17 references
8      public class Person
9      {
10         10 references
11         public int Id { get; set; }
12         5 references
13         public string Fornavn { get; set; }
14         5 references
15         public string Etternavn { get; set; }
16         5 references
17         public string Telefon { get; set; }
18         8 references
19         public string Email { get; set; }
20     }
21 }

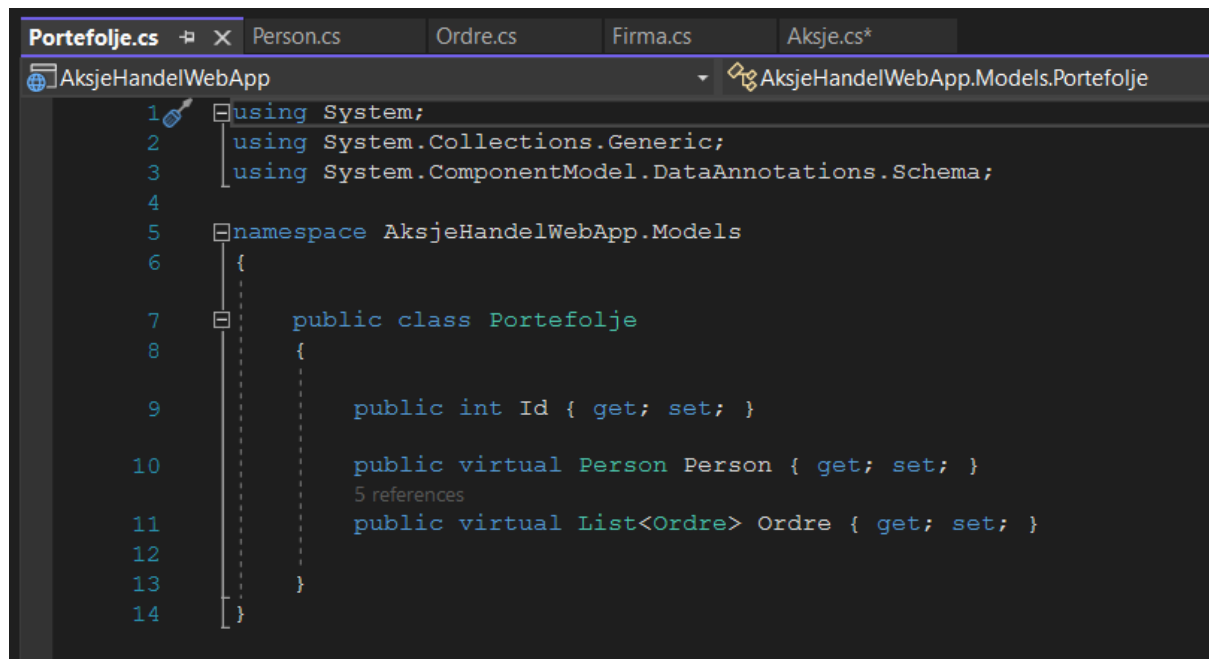
```

Portefølje-tabell:

Porteføljen kobles til en person-ID med portefølje-ID som primærnøkkel.

Table:  Portefoljer 

	Id	PersonId
	Filter	Filter
1	1	1



```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations.Schema;
4
5 namespace AksjeHandelWebApp.Models
6 {
7     public class Portefolje
8     {
9         public int Id { get; set; }
10
11         public virtual Person Person { get; set; }
12         5 references
13         public virtual List<Ordre> Ordre { get; set; }
14     }
15 }
```

Hva er CRUD og hvordan har vi implementert CRUD?

CRUD står for Create, Read, Update og Delete som er de fire grunnleggende operasjonene for vedvarende lagring. På vår webapplikasjon kan man lage ny bruker og legge inn nye bestillinger. Man kan lese opplysninger knyttet til eksisterende brukere, bestillinger og aksjer gjennom databasen. Det er mulig å oppdatere brukerinformasjon gjennom applikasjonen, i tillegg til at aksjeverdiene i databasen oppdateres dynamisk. Man kan også slette en bruker og fjerne all tilhørende informasjon.

Hvordan er systemet bygget opp?

I PersonRepository behandles dataene i databasen.

hentPortefolje() kobler porteføljen opp med tilhørende bruker-ID som vist under.

```

2 references
public class PersonRepository : IPersonRepository
{
    private readonly DB _db;

    0 references
    public PersonRepository(DB db)
    {
        _db = db;
    }

    2 references
    public async Task<Portefolje> hentPortefolje(int id)
    {
        try
        {
            Portefolje enPortefolje = _db.Portefoljer.Single(x => x.Person.Id.Equals(id));
            return enPortefolje;
        }
        catch
        {
            return null;
        }
    }
}

```

hentAksjer() henter aksjene som ligger i databasen, med tilhørende data.

```

2 references
public async Task<List<Aksje>> hentAksjer()
{
    try
    {
        List<Aksje> alleAksjer = await _db.Aksjer.Select(a => new Aksje
        {
            Id = a.Id,
            Verdi = a.Verdi,
            Firma = a.Firma
        }).ToListAsync();

        return alleAksjer;
    }
    catch
    {
        return null;
    }
}

```

Funksjonen hentPerson() henter ut data til en bruker i databasen basert på bruker-ID.

2 references

```
public async Task<Person> hentPerson(int id)
{
    try
    {
        Person enPerson = await _db.Personer.FindAsync(id);
        return enPerson;
    }
    catch
    {
        return null;
    }
}
```

sjekkPerson() sjekker e-posten opp i mot bruker-ID for å validere en bruker.

2 references

```
public async Task<int> sjekkPerson(string email)
{
    try
    {
        Person enPerson = _db.Personer.First(x => x.Email == email);
        return enPerson.Id;
    }
    catch
    {
        return 0;
    }
}
```

lagrePerson() registrerer en ny bruker, og legger brukeren en ID, portefølje og kobler brukeren opp til en e-post.

```

2 references
public async Task<int> lagrePerson(Person innPerson)
{
    try
    {
        var nyPortefolje = new Portefolje();
        nyPortefolje.Person = innPerson;
        _db.Portefoljer.Add(nyPortefolje);
        await _db.SaveChangesAsync();
        Person enPerson = _db.Personer.First(x => x.Email == innPerson.Email);
        return enPerson.Id;
    }
    catch
    {
        return 0;
    }
}

```

registrerOrdre() registrerer en ny ordre, både kjøp og salg. Dette gjøres ved hjelp av en boolean verdi, hvor type 0 er salg og 1 er kjøp.

```

2 references
public async Task<bool> registrerOrdre(Ordre innOrdre)
{
    try
    {
        var nyeOrdre = new List<Ordre>();
        innOrdre.Kjøpspris = innOrdre.Aksje.Verdi * innOrdre.AntallAksjer;
        nyeOrdre.Add(innOrdre);
        Portefolje enPortefolje = await _db.Portefoljer.FindAsync(innOrdre.Portefolje.Id);
        enPortefolje.Ordre = nyeOrdre;
        await _db.SaveChangesAsync();

        return true;
    }
    catch
    {
        return false;
    }
}

```

hentFirma() henter firmaet fra databasen basert på firma-IDen.

```

2 references
public async Task<Firma> hentFirma(int id)
{
    try
    {
        Firma etFirma = await _db.Firmaer.FindAsync(id);
        return etFirma;
    }
    catch
    {
        return null;
    }
}

```

Funksjonen slettBruker() finner en bruker basert på bruker-ID og fjerner all tilhørende data inkludert porteføljen.

```

2 references
public async Task<bool> slettBruker(int id)
{
    try
    {
        Person enPerson = await _db.Personer.FindAsync(id);
        Portefolje enPortefolje = await _db.Portefoljer.FirstAsync(x => x.Person.Id == id);
        _db.Personer.Remove(enPortefolje.Person);
        foreach (Ordre ordre in enPortefolje.Ordre)
        {
            _db.Ordre.Remove(ordre);
        }

        _db.Portefoljer.Remove(enPortefolje);

        await _db.SaveChangesAsync();
        return true;
    }
    catch
    {
        return false;
    }
}

```

hentAksje() henter en aksje fra databasen etter aksje-ID.

2 references

```
public async Task<Aksje> hentAksje(int id)
{
    try
    {
        Aksje enAskje = await _db.Aksjer.FindAsync(id);
        return enAskje;
    }
    catch
    {
        return null;
    }
}
```

visPortefolje() henter ut dataene knyttet til en portefolje. Den regner ut antall aksjer ut i fra hvor mange som er kjøpt og solgt i historikken. Funksjonen lager en liste av aksjene, og legger til en ny rad for hver ny type aksje den finner. Verdiene settes til to desimaler.

```

2 references
public async Task<List<VisPortefolje>> visPortefolje(int id)
{
    try
    {
        List<VisPortefolje> nyVisning = new List<VisPortefolje>();
        Portefolje enPortefolje = _db.Portefoljer.First(x => x.Person.Id == id);
        bool sjekk = false;
        foreach (Ordre s in enPortefolje.Ordre)
        {
            foreach (VisPortefolje enport in nyVisning)
            {
                if (enport.Aksje.Id == s.Aksje.Id)
                {
                    sjekk = true;
                    if (s.Type)
                    {
                        enport.Antall = enport.Antall + s.AntallAksjer;
                    }
                    else
                    {
                        enport.Antall = enport.Antall - s.AntallAksjer;
                    }
                }
            }
        }
        if (sjekk == false)
        {
            var NyLinje = new VisPortefolje();
            NyLinje.Aksje = s.Aksje;
            if (s.Type)
            {
                NyLinje.Antall = s.AntallAksjer;
            }
            else
            {
                NyLinje.Antall = 0 - s.AntallAksjer;
            }
            nyVisning.Add(NyLinje);
        }
        sjekk = false;
    }
    List<VisPortefolje> port = new List<VisPortefolje>();
    foreach (VisPortefolje enport in nyVisning)
    {
        if (enport.Antall > 0)
        {
            enport.Verdi = enport.Antall * enport.Aksje.Verdi;
            enport.Verdi = (float)Math.Round(enport.Verdi * 100f) / 100f;
            port.Add(enport);
        }
    }
    return port;
}
catch
{
    return null;
}
}

```

endrePerson() endrer brukeropplysninger basert på bruker-IDen.

```

0 references
public async Task<bool> endrePerson(Person innPerson)
{
    try
    {
        var endreObjekt = await _db.Personer.FindAsync(innPerson.Id);
        endreObjekt.Fornavn = innPerson.Fornavn;
        endreObjekt.Etternavn = innPerson.Etternavn;
        endreObjekt.Email = innPerson.Email;
        endreObjekt.Telefon = innPerson.Telefon;
        await _db.SaveChangesAsync();
    }
    catch
    {
        return false;
    }
    return true;
}

```

hentOrdre() henter ut en brukers ordre som vises i handelshistorikken på *Min side*.

```

2 references
public async Task<Portefolje> hentOrdre(int id)
{
    try
    {
        Portefolje enPortefolje = await _db.Portefoljer.SingleAsync(x => x.Person.Id.Equals(id));
        return enPortefolje;
    }
    catch
    {
        return null;
    }
}

```

oppdaterBors() oppdaterer aksjeprisene fortløpende for å skape dynamikk, og muliggjør å lage en egen modul for *Avkastning*.

```

2 references
public async Task<List<Aksje>>oppdaterBors()
{
    Random rnd = new Random();
    List<float> random = new List<float>();
    random.Add((float)1.009);
    random.Add((float)1.008);
    random.Add((float)1.007);
    random.Add((float)1.006);
    random.Add((float)1.005);
    random.Add((float)1.004);
    random.Add((float)1.003);
    random.Add((float)1.002);
    random.Add((float)1.001);
    random.Add((float)0.999);
    random.Add((float)0.998);
    random.Add((float)0.997);
    random.Add((float)0.996);
    random.Add((float)0.995);
    random.Add((float)0.994);
    random.Add((float)0.993);
    random.Add((float)0.992);
    random.Add((float)0.991);
    List<Aksje> alleAksjer = await _db.Aksjer.ToListAsync();
    foreach (Aksje aksje in alleAksjer)
    {
        aksje.Verdi = aksje.Verdi * random.ElementAt(rnd.Next(random.Count()));
        aksje.Verdi = (float)Math.Round(aksje.Verdi * 100f) / 100f;
        await _db.SaveChangesAsync();
    }
    return alleAksjer.ToList();
}

```

kjøptFor() beregner til hvilken pris en aksje er kjøpt.

```

2 references
public async Task<float> kjøptFor(int id)
{
    try
    {
        List<VisPortefolje> nyVisning = new List<VisPortefolje>();
        Portefolje enPortefolje = _db.Portefoljer.First(x => x.Person.Id == id);
        float pris = 0;
        foreach (Ordre s in enPortefolje.Ordre)
        {
            if (s.Type)
            {
                pris = pris + s.Kjøpspris;
            }
            else
            {
                pris = pris - s.Kjøpspris;
            }
        }
        return pris;
    }
    catch
    {
        return 0;
    }
}

```

DBInit

I DBInit har vi lagt til noen verdier som legges inn i programmet før det startes opp. Her legges det inn en bruker, samt portefølje og ordre knyttet til denne brukeren. I tillegg legges det inn noen firmaer og tilhørende aksjer inn i databasen. Ved programstart blir databasen slettet og bygd opp på ny. I denne versjonen av programmet har vi valgt og slette databasen ved oppstart for å holde databasen ryddig. Dette blir endra på et senere tidspunkt.

Tilleggsfunksjonalitet

Vi har gjort det mulig å lage en bruker og logge inn (foreløpig uten *log-in session*), hvor kjøp og salg av aksjer blir knyttet til bruker-IDen. Aksjeverdiene oppdateres dynamisk ved et tidsintervall på 10 sekunder. Dette gir også muligheten til å ha en modul for avkastning, noe vi har lagt til på vis portefølje.

For å holde kontroll på de ulike brukerne har vi valgt å bruke `storageSessions()`. Når en bruker logger inn blir e-posten sjekket opp mot brukere lagret i databasen. Hvis e-posten allerede eksisterer i databasen vil serversiden returnere bruker-IDen. Denne vil bli lagret i en `storageSession` på klientsiden.

Konklusjon:

Ved hjelp av strukturert arbeid og effektiv fordeling av arbeidsoppgaver har gruppen klart å skape et MVP til en aksjehandelswebapplikasjon som svarte på brukerkravene. Dette har gitt oss en god innføring i hvordan å sette opp både front- og back-end ved hjelp av .NET Core.

Kilder:

Ikoner:

- "Favicon" har vi hentet gratis fra <https://icons8.com/>
- "Login icon" har vi hentet gratis fra <https://www.flaticon.com/free-icons/login>
- "User icon" har vi hentet gratis fra <https://www.flaticon.com/free-icons/login>