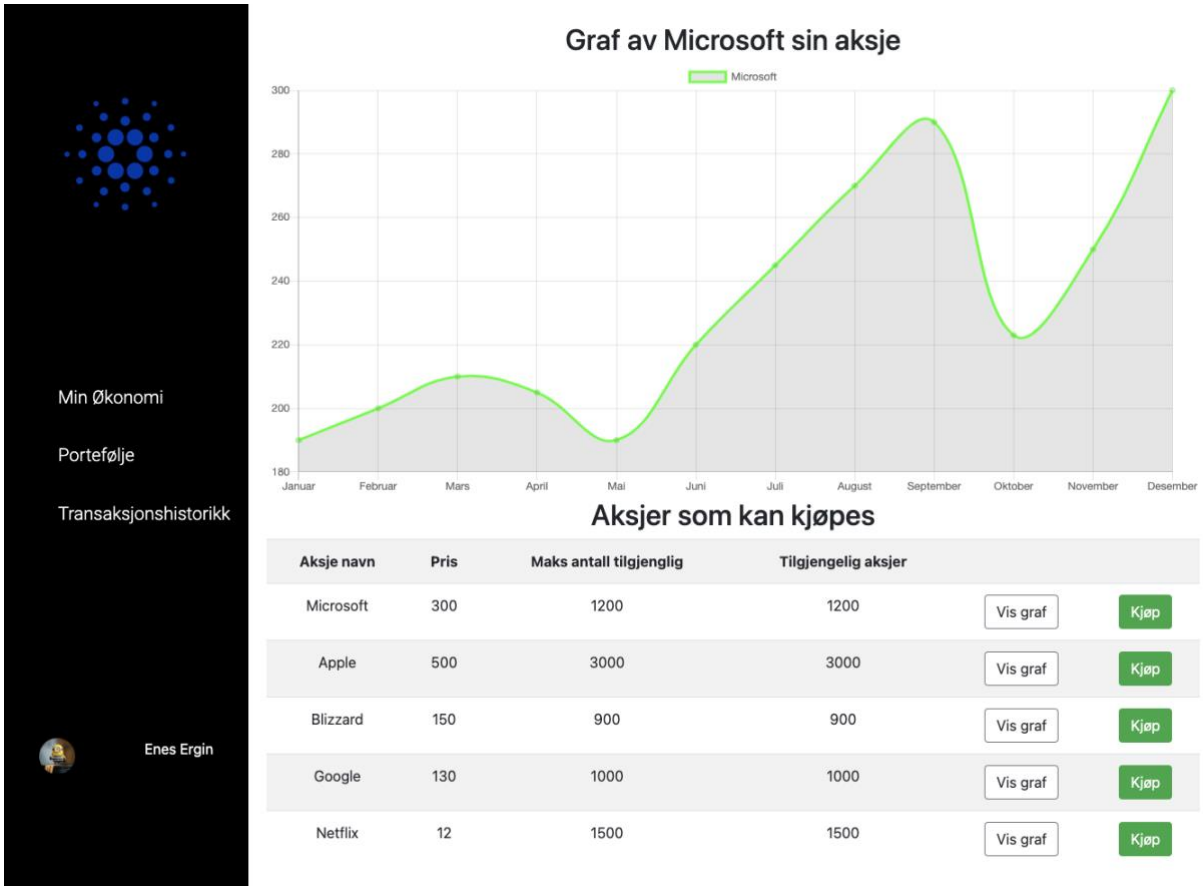
Gruppeoppgave 1 - Dokumentasjon

ITPE3200 – Webapplikasjoner



Zubeyr Mustafa Ahmed – s356342, Tin Tun Naing – s354358

Shahvez Mahmood – s354404, Harman Singh Tathgar – s351940

Enes Ergin – s351880

03.10.2022

Innholdsfortegnelse

Introduksjon

I dette prosjektet skal vi ta for oss en webapplikasjonsoppgave som er basert på en av tre mulige oppgaver. En felles beslutning ble tatt, og aksjeoppgaven ble enstemmig valgt. Før prosjektstart hadde alle medlemmer sett alle aktuelle moduler og forelesningsnotater på egen hånd, slik at vi hadde felles grunnkunnskap.

Kontekst for prosjektet

Prosjektbeskrivelsen sier at for dette prosjektet er det nokså åpent, men samtidig har den en rekke krav og funksjonaliteter som må iverksettes. Først og fremst går prosjektet ut på å lage en webapplikasjon der man kan se på, selge og kjøpe aksjer. Det viktigste kravet er å sørge for en Minimum Viable Product (MVP), som viser at applikasjonen er teknisk mulig. Resten av kravene til selve oppgaven består av å bruke riktig .Net Core versjon og vise kunnskap i det og EF.

Oppgaven hadde også en rekke funksjonaliteter som skulle oppfylles. Et interaktivt brukergrensesnitt i form av en nettside, i tillegg til at ikke trenger å ha en bruker til å logge inn. CRUD er også et krav som blir vektlagt, og prosjektet ble gjort med disse kravene og funksjonalistene i bakhodet.

Med tanke på at Gruppeoppgave 2 skal bygge videre på denne løsningen, har det vært en del funksjonaliteter og bedre design som skal brukes i det neste prosjektet. En av disse er å mer enn en bruker i løsningen. Derfor har vi forsøkt å lage en applikasjon som kan skaleres kun ved noen få endringer.

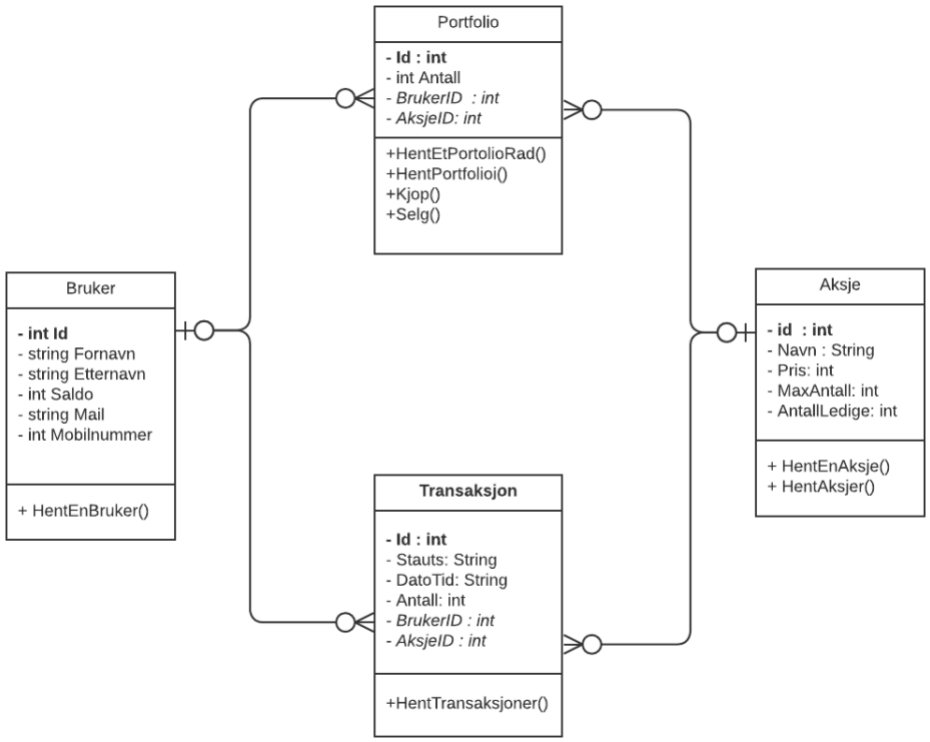
Siden denne løsningen også bare har en bruker, har vi ikke tatt bruk av et API og bare hardkodet verdiene for ulike aksjer. Noen av funksjonene vil naturlig hadde trengt en brukerID, vil ikke trenge å finne bruker.

Relasjonsdatabase og klassediagram

Under utvikling av hvilket som helst produkt eller system som skal brukes av kunder, er det viktig å kunne ha en oversikt over hva kunden kan gjøre. Dette gjorde vi ved å lage ulike use-cases, som gir oss ideer til hvordan systemet skal oppføre seg. Disse var «bruker skal kunne kjøpe aksjer», «bruker skal kunne selge aksjer», «bruker skal kunne se ordrehistorikken sin» og «bruker skal kunne se tilgjengelige aksjer på markedet». Ordrehistorikken var ikke en del av oppgavebeskrivelsen, men vi mente at det var en sentral del av en aksjeapplikasjon.

Neste steg i utviklingen var å sette opp relasjonsdatabaser. Hensikten med dette er å gjøre det mer oversiktlig og lage logiske forhold mellom to tabeller. Senere når man skal lage funksjoner som henter data fra ulike tabeller, vil det gjøre det anstrengende å finne riktige objekter og attributter. Dette kan også skade strukturen vår og logikken kan bli vanskeligere å iverksette dersom dette ikke gjøres må en god måte.

I denne delen av prosessen, startet vi tidlig med å teste ut ulike databaser og koblingene mellom disse. Det var viktig å vite hvilken entitet som skulle utføre CRUD (Create, Read, Update, Delete), siden dette var et av kravspesifikasjonene.



Funksjonene som befinner seg i klassediagrammet, men det betyr ikke at funksjonene er i den klassen. Det er kun for å vise i hvilken sammenheng de blir brukt. Som nevnt tidligere, blir CRUD gjennomført på denne måten;

C (Create); denne delen av CRUD utføres dersom man kjøper en aksje når man ikke eier aksjer, og vi får et nytt objekt i porteføljе tabellen.

R (Read); dette blir utført når man skal se hva man har i porteføljen sin.

U (Update); det er to ulike måter porteføljen blir oppdatert. Disse er ved hjelp av kjøpAksje og selgAksje.

D (Delete); dette skjer når man selger alle sine aksjer, da vil det fjernes fra porteføljen.

Løsning

Frontend

På frontend-delen av prosjektet har vi totalt fem sider. Vi har en navigasjonsbar til venstre som viser alle sider det skal være mulig å navigere, uansett hvor man befinner seg i applikasjonen. Det er tre sider i navigasjonsbaren, i tillegg er det også kjøps- og salgssiden som man kan aksessere gjennom dem.

Den første siden i den er Min Økonomi, som virker som en hjemmeside i applikasjonen. Her har man tilgang til å kjøpe aksjer og se aksjeinformasjon som navn, pris per aksje, maks antall tilgjengelige og antall ledige aksjer. I tillegg har vi et interaktivt diagram som viser verdien til en aksje over 12 måneder. Denne kan byttes ved å trykke på "Vis graf"-knappen som står hos hver aksje.

Den andre siden i navigasjonsbaren er Portefølje. Her har man oversikt over alle egne aksjer man eier; altså pris per aksje, beholdning, og sum (verdi). På denne siden har man to knapper hos hver aksje man eier, disse er "Kjøp mer" og "Selg", som tar deg med til de respektive sidene. Dersom man ikke eier noen aksjer, vil man ikke kunne få opp disse mulighetene. Øverst til høyre i porteføljen har man egen saldo.

Som nevnt tidligere, kan man aksessere kjøpssiden via Min Økonomi og Porteføljen (hvis man allerede eier den aksjen fra før, ellers står den tom). Når man trykker på "Kjøp"- eller "Kjøp mer"-knappene, er aksjen man skal kjøpe allerede valgt. Det vil si at man har bestemt hvilken aksje som skal kjøpes før man kommer inn på den siden. På kjøpssiden har man to

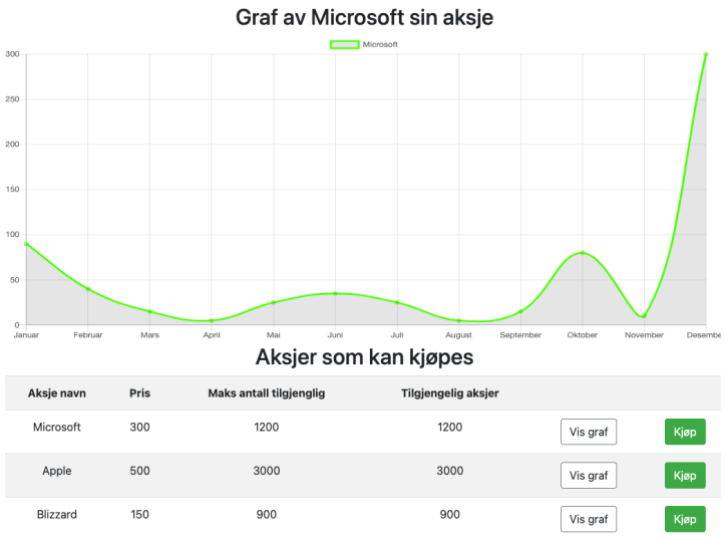
bokser, hvor den til venstre viser aksjenavn, pris per aksje og antall ledige/maks antall. Boksen til høyre inneholder et inputfelt hvor man taster inn antall aksjer man ønsker å kjøpe, og en bekreft knapp for dette. I tillegg er det en “Se porteføljen min”-knapp.

Det er kun en måte man kan aksessere salgssiden på, og dette er gjennom Portefølje. Siden det ikke er mulig å selge aksjer man ikke har, er det naturlig at man må navigere først til Portefølje for å finne salgssiden. Som nevnt tidligere, har alle aksjer man eier i Portefølje hver sin "Selg”-knapp. Når man trykker på denne, er oppsettet på siden er ganske likt kjøpssiden, med to bokser, en med aksjeinfo og en annen med input. På inputfeltet må man taste inn antall aksjer man vil selge, også bekrefte. Her er det også en “Se porteføljen min”-knapp.

Den siste siden i navigasjonsbaren er Transaksjoner, som lister opp alle transaksjoner som har foregått. Her får vi informasjon som hva slags transaksjonstype (kjøp eller salg), dato og tid, aksjenavn, pris per aksje, antall aksjer og sum.

Ekstra funksjonaliteter

Før vi startet prosjektet, hadde vi alltid forhåpninger om å få til ekstra funksjonaliteter. Dette endte med at vi lagde en liste over ekstra funksjoner på webapplikasjonen som kan være et greit å ha med. Et av disse var å ha en graf, som viser verdien til en aksje over lenger tid. Se bilde under.



Det er en interaktiv graf som ble hentet via chart.js biblioteket fra chartjs.org. Koden til grafen ble skrevet ved hjelp av dokumentasjonen til nettsiden.

En annen implementasjon som ikke var en del av oppgaven var å ha en ordrehistorikk, eller transaksjoner. Dette mente vi var en naturlig del av en aksjeapplikasjon, og hadde planlagt dette fra tidlig.

Kode

Klasser

```
public class Bruker
{
    public int Id { get; set; }
    public string Fornavn { get; set; }
    public string Etternavn { get; set; }
    public int Saldo { get; set; }
    public string Mail { get; set; }
    public int Mobilnummer { get; set; }
}

public class Aksje
{
    public int Id { get; set; }
    public string Navn { get; set; }
    public int Pris { get; set; }
    public int MaxAntall { get; set; }
    public int AntallLedige { get; set; }
}
```

```
public class Transaksjon
{
    public int Id { get; set; }
    public string Status { get; set; }
    public string DatoTid { get; set; }
    public int Antall { get; set;}

    //Fra Aksje
    public int AksjeId { get; set; }
    public string AksjeNavn { get; set; }
    public int AksjePris { get; set; }

    //Fra Bruker
    public int BrukerId { get; set; }
}

public class Portfolio
{
    public int Id { get; set; }
    public int Antall { get; set; }

    //Info fra Aksje
    public int AksjeId { get; set; }
    public string AksjeNavn { get; set; }
    public int AksjePris { get; set; }

    //Info fra Bruker
    public int BrukerId { get; set; }
}
```

Funksjoner

Det er åtte ulike metoder som skal benyttes, og det er disse metodene som setter liv i løsningen.

Som nevnt tidligere, er det ikke behov for flere brukere, så derfor er det manuelt satt id = 1 i noen av funksjonene.

F. eks.

```
{
    Brukere enBruker = await _db.Brukere.FindAsync(1);
    var hentetBruker = new Bruker()
    {
```

lagTransaksjon;

Denne funksjonen kalles på kjøp- og salgsfunksjonene. Den tar brukerID, aksjeinfo, status (om det kjøps eller salg) og lagrer dette i Transaksjonsdatabasen. I tillegg legger den til dato og tid til når funksjonen blir kalt på, altså hver gang man kjøper eller selger.

Selg;

Denne funksjonen finner først hvilken aksje som skal selges fra aksjeID. Deretter sjekker den om han prøver å selge mer enn det man eier, og om man prøver å selge null. Hvis man selger delvis det man eier av en aksje, vil saldoen gå opp, antall aksjer du solgte vil trekkes fra hva de eier og antall ledige aksjer vil gå opp. Deretter vil den kalle på lagTransaksjon-funksjonen.

Det er også en sjekk som kontrollere om man prøver å selge alle aksjene sine, da vil selve raden i porteføljen slettes i tillegg.

Kjøp;

Denne funksjonen finner først hvilken aksje det er som skal kjøpes. Deretter sjekker den om man har nok saldo til å utføre kjøpet, også sjekker den om man prøver å kjøpe et lovlig antall aksjer. Det vil si at den sjekker om kjøper mer enn antall ledige aksjer og mer enn 0 aksjer.

Siden man skal legge til en rad i sin portefølje, vil funksjonen også sjekker om bruker allerede har den aksjen, for da trenger bare antallet, saldo og antall ledige endre seg. Hvis man ikke eier den aksjen fra før av, vil funksjonen legge til en rad i porteføljen og sette inn verdiene.

Til slutt vil saldo trekkes fra summen og antall man har kjøpt trekkes fra antall ledige.

HentEnBruker;

Siden vi kun har en bruker i løsningen, trenger vi ikke å ta in en brukerID. Vi setter den som 1 med en gang. Så returnerer den brukeren.

HentEnAksje;

I denne funksjonen finner vi først hvilken aksje det gjelder via id, og returnerer aksjeobjektet.

HentEtPortfolioRad;

I denne funksjonen finner vi først aksje i porteføljen det gjelder via id, og returnerer den porteføljeraden (aksjen i porteføljen).

HentAksjer;

I denne funksjonen blir alle aksjene i aksjedatabasen returnert.

HentPortfolio;

I denne funksjonen blir alle objektene i porteføljen returnert.

HentTransaksjoner;

I denne funksjonen blir alle tidligere kjøp og salg listet opp.

Ekstra funksjonaliteter (Kode)

```
function endreChart(navn, i) {

    $("#grafNavn").html("Graf av " + navn + " sin aksje");

    const labels = [
        "Januar",
        "Februar",
        "Mars",
        "April",
        "Mai",
        "Juni",
        "Juli",
        "August",
        "September",
        "Oktober",
        "November",
        "Desember"];

    const data = [
        {
            labels: labels,
            datasets: [
                {
                    label: "Microsoft",
                    borderColor: "rgb(0,255,0)",
                    data: [190, 200, 210,205, 190, 220, 245, 270, 290, 223, 250, 300
                }
            ]
        },
        {
            labels: labels,
            datasets: [
                {
                    label: "Apple",
                    borderColor: "rgb(220,20,60)",
                    data: [90, 100, 120, 95, 150, 230, 280, 300, 45, 470, 510, 500]
                }
            ]
```

Som nevnt tidligere, ble har vi implementert chart.js biblioteket for grafen og koden er skrevet ved hjelp av dokumentasjonen på chartjs.org.

Først har vi en overskrift som endrer seg basert på hvilken «Vis graf»-knapp man trykket på. Deretter setter funksjonen de ulike verdiene som skal vises i x- og y-aksene, og disse er naturligvis pris per aksje og måned. Vi hardkodet verdiene for hver av de aksjene. Verdiene til aksjene vil da ikke aktivt endre på seg.

Til slutt vil funksjonen ved hjelp av denne koden, lage grafen og sette den i Dashboard, som er en canvas i index.html.

```
const input = parseInt(i)

const config = {
    type: "line",
    data: data[input - 1],
    options: {},
};

const myChart = new Chart(document.querySelector(".dashboard"), config);
```

Transaksjoner var en funksjon som ble utført hver gang det skjedde et suksessfullt kjøp eller salg av aksjer. Disse ble lagt i en database, som blir formattert og fylt ut når man åpner transaksjonshistorikk.

Kilder

Chart: https://www.chartjs.org/docs/latest/samples/line/point-styling.html