**Prosjektoppgave for Webapplikasjoner Del 1, 2022**

Gruppe: 4200

Kandidatnummere:

284

191  
204  
288

ITPE3200

Table of Contents

[Introduksjon: 3](#_Toc506339018)

[Applikasjon beskrivelse: 3](#_Toc1157814684)

[Design Prototype: 4](#_Toc1191684996)

[Skisser: 4](#_Toc1057582783)

[Første prototype (28.09.2022): 4](#_Toc1011903688)

[Ulike prototyper for Kjøp og selg html-en (10.07.2022): 6](#_Toc1466220079)

[Besluttet Design: 8](#_Toc767678859)

[Index.html: 8](#_Toc1082890418)

[NB : 10](#_Toc1809343122)

[Kjøp.html 12](#_Toc1600058123)

[Sell.html 13](#_Toc1935190085)

[- Diagrammer 15](#_Toc299603403)

[MVP (Minimal viable product): 16](#_Toc494219991)

[Index.html: 16](#_Toc1908398579)

[Purchase.html: 17](#_Toc910962417)

[Sell.html: 17](#_Toc1333771796)

[Kode beskrivelse: 17](#_Toc343926602)

[Oppsettet: 17](#_Toc1766563559)

[Controllers: 18](#_Toc1447506334)

[- StockPurchaseController.cs: 18](#_Toc914542537)

[Model: 20](#_Toc601096430)

[- Stock.cs og StockPurchase.cs: 20](#_Toc1313029273)

[- StockDB.cs: 20](#_Toc1911924235)

[wwwroot: 21](#_Toc105880486)

[- index.html: 21](#_Toc652687893)

[- purchase.html: 24](#_Toc202457187)

[- sell.html: 27](#_Toc1803048090)

[- Script.js: 30](#_Toc206590849)

[Konklusjon: 37](#_Toc26219286)

[Referanser: 38](#_Toc1407207862)

[Youtube: 38](#_Toc1648013641)

[Akjsenettsider: 38](#_Toc416469949)

[Api: 38](#_Toc2137524289)

# Introduksjon:

I denne prosjektoppgaven har vi fått en oppgave med å implementer en side for å kunne se, kjøpe og selge aksjer. Prinsippet med oppgaven er å vise fram forståelsen av .NET Core og Entity Framework som ble lært bort i programmet Visual Studio. Men samtidig vise en mestring av programspråkene HTML, CSS, JavaScript og C#. Oppgaven ber om at kunden visuelt kan se hvor man kan kjøpe aksjer og selge aksjer, i tillegg til at funksjonaliteten også er satt fram.

# Applikasjon beskrivelse:

Frontend delen av nettsiden er en MPA (multiple page application) basert på JavaScript. Vi har valgt dele nettsiden inn i 3 sider: en forside, en kjøp side og en salg side. Ettersom det ikke var et stort fokus på design så valgte vi å benytte Bootstrap for designet. Bootstrap er et CSS bibliotek som dekker de fleste design behov og har mange ferdiglagde komponenter som tabeller og knapper. Dette gjorde at nettsiden så bra ut med minimalt arbeid som førte til at vi kunne fokusere på funksjonalitet. I tillegg til dette er sidene responsive slik at siden blir justert etter skjerm størrelse og orientering.

Forsiden viser en oversikt over hvilke aksjer som er tilgjengelig å kjøpe i form en av en tabell. Innholdet i denne tabellen blir dynamisk fylt inn av JavaScript med aksjedata som er hentet fra .NET APIen. Den andre tabellen på forsiden kombinerer data fra aksjene og aksje kjøpene fra databasen for å fremvise hvilke aksjer som er kjøpt. Radene i tabellen bruker aksje- og aksjekjøp IDene til å linke direkte til en ressurs på kjøp og salg sidene. Det vil si at når man trykker på knappen for å kjøpe en aksje, så vil den aksjen være valgt på neste side.

Kjøp siden henter den samme aksje dataen fra APIen og fremstiller dette i et input element hvor kunden kan enkelt velge hvilken aksje de vil kjøpe. Hvis brukeren navigerte til kjøp siden ved å trykke på en av kjøp knappene i tabellen på forsiden, så vil aksjen automatisk være forhåndsvalgt i nedtrykksmenyen. Deretter velger kunden mengden aksjer de ønsker å kjøpe i boksen under. Når de trykker på “Confirm purchase” blir en POST forespørsel med kjøpsdata sendt til APIen i form av et JSON objekt. I .NET applikasjonen har vi en API kontroller for aksjekjøp sin behandler forespørselen og lagrer kjøpet i en SQLite database. Hvis alt var vellykket, blir kunden omdirigert tilbake til forsiden hvor tabell dataen blir igjen hentet på nytt for å reflektere det nye kjøpet.

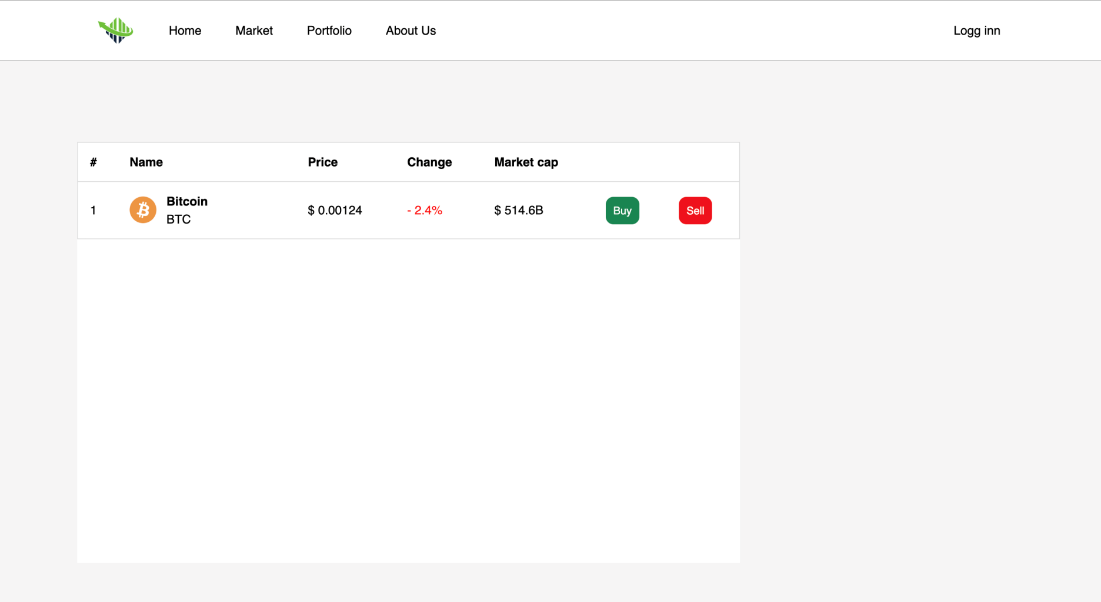
Den siste siden er for å selge aksjene man har kjøpt. Her velger man hvilke aksjer, og hvor store mengder av den aksjen man ønsker å selge. På samme måte som kjøp siden blir data fra APIen hentet og fyller inn innholdet på siden dynamisk. Før forespørselen blir sendt, så validerer JavaScript at kunden har nok mengde av aksjen de ønsker å selge. Hvis man prøver å selge men enn man har, så får man en feil melding.

Oppgaven ber om minst en CRUD implementasjon. Dette var noe vi fikk vansker med siden vi ikke fant et element der CRUD vil være fornuftig, spesielt oppdateringsdelen av CRUD. For å oppfylle kravet bestemte vi oss for å gjøre aksjekjøp om til CRUD der man kan opprette et kjøpt aksje, lese aksje på hjemmesiden, selge aksje, og oppdatere mengde du har kjøpt.

# Design Prototype:

## Skisser:

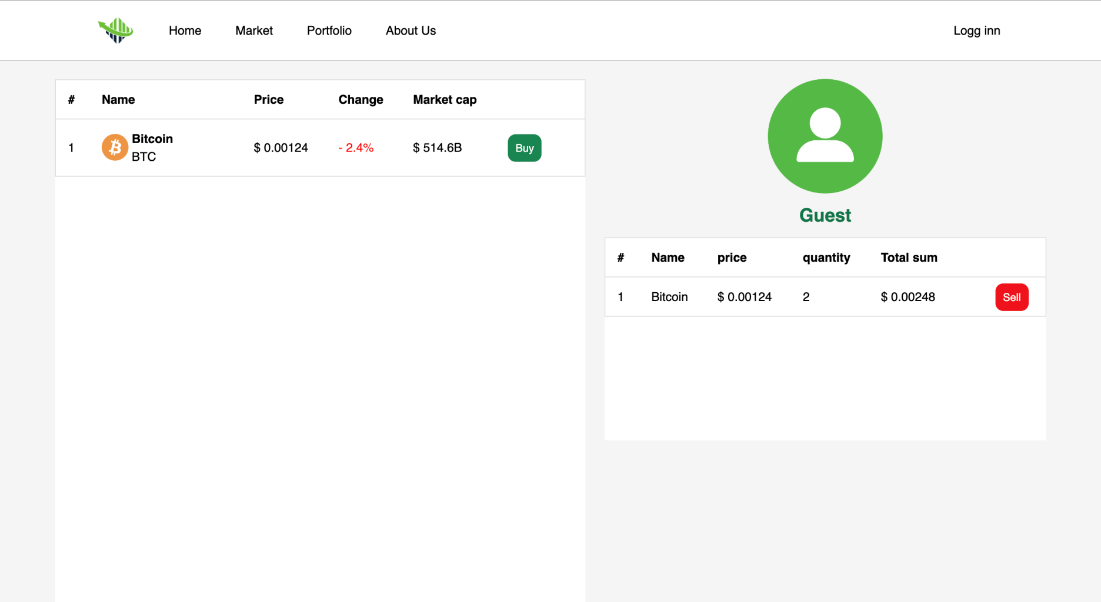
### **Første prototype (28.09.2022):**



Figur 1 - Første prototype

Første prototypen ble lagd i fokus for en generell enkel aksjeside som er lett å implementere. Med fokus av å implementere en side for å kunne se, kjøpe og selge aksjer i det mest brukervennlige måten. Inspirasjonen for prototypen kom fra ulike aksje-nettsider som for eksempel Coinbase, Tradingview og investor.dn.

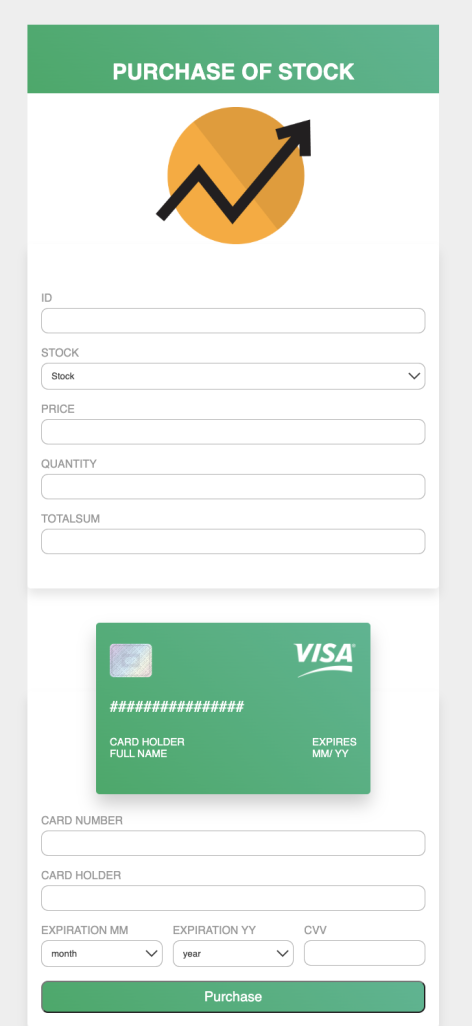
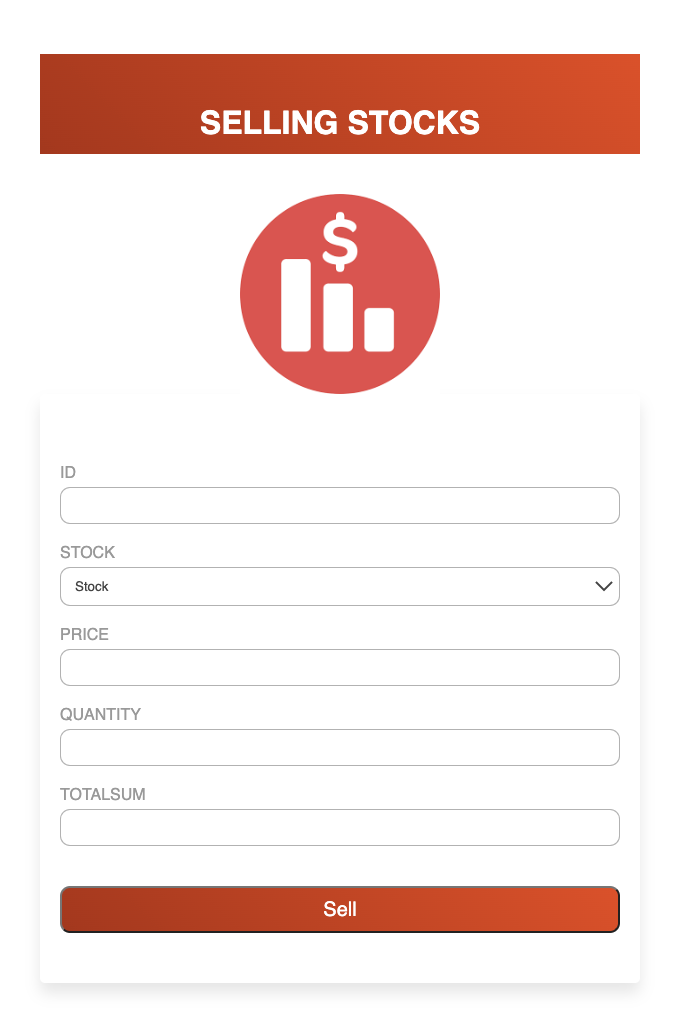
**Andre prototype (29.09.2022):**



Figur 2 - Oppdatert prototype

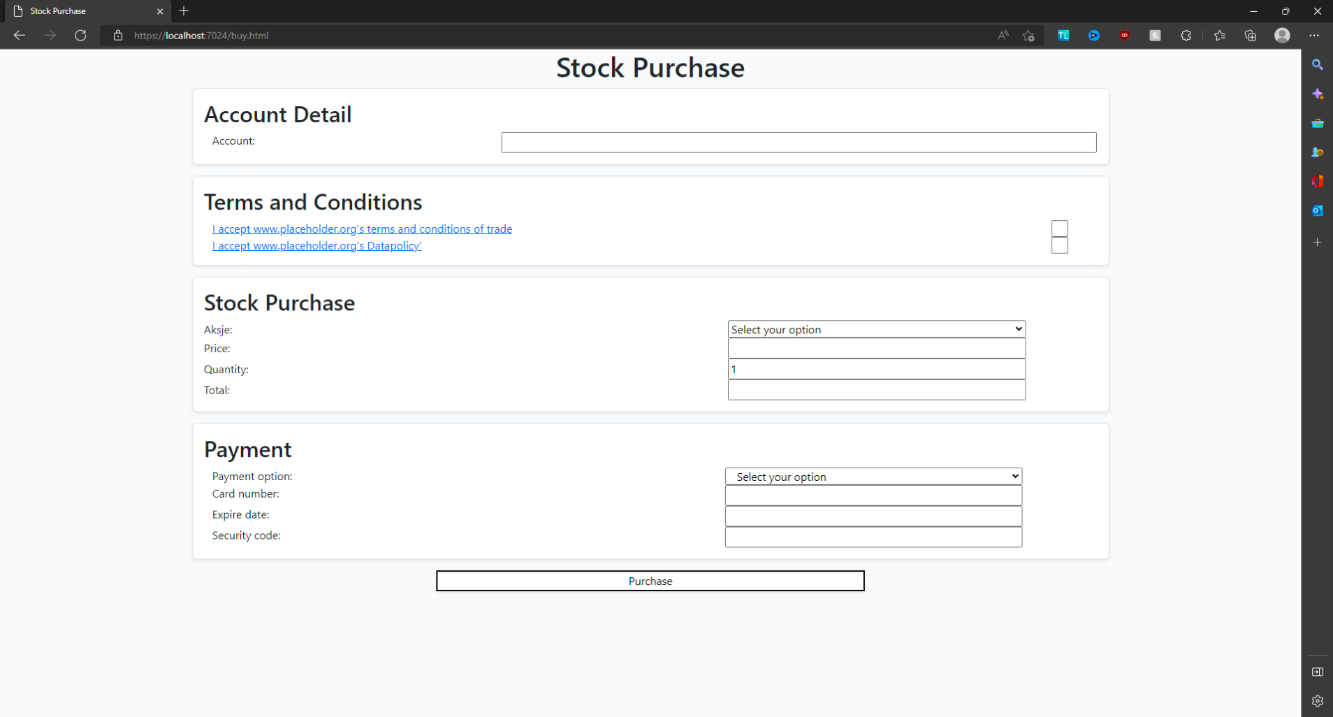
Andre prototypen gikk mer i dybden av første prototypen med tanken over del 2 av prosjektoppgaven. Vi ble informert om at del 2 vil kunne ha innlogging og registrering som element, og derfor tenkte vi å utvikle en prototype som enkelt kunne inneholde begge deler uten å gjøre noen vesentlige justeringer.

### **Ulike prototyper for Kjøp og selg html-en (10.07.2022):**

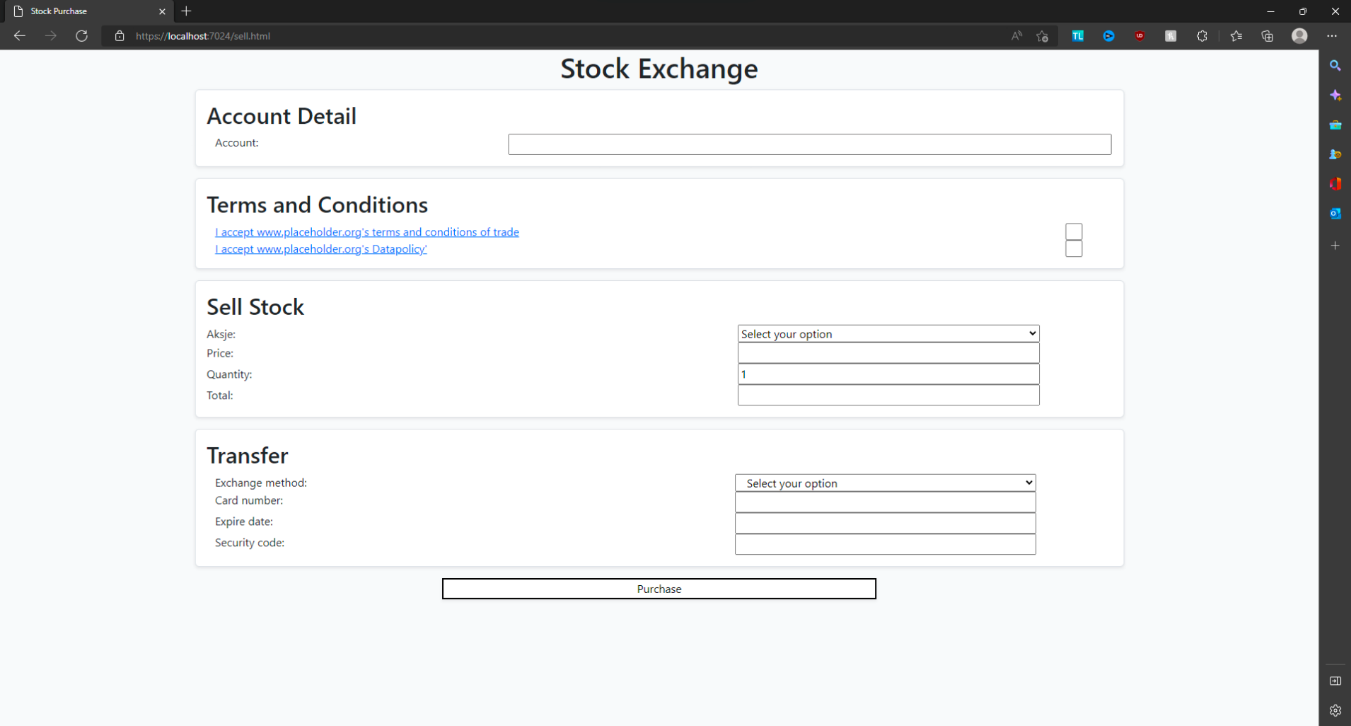


Figur 3 - Kjøpsside prototype. Figur 4 – Salgsside prototype.

En kjøp og salg format som konsentrerte seg om en enkel og brukervennlig design var temaet i figur 3 og 4. Mens den tydelig viser hvilken informasjon som skal inkluderes i tekstformat, viser den også en visuell div-container som ligner på et visa-kort og viser tilleggsinformasjon i tilfelle kunden er uklar.

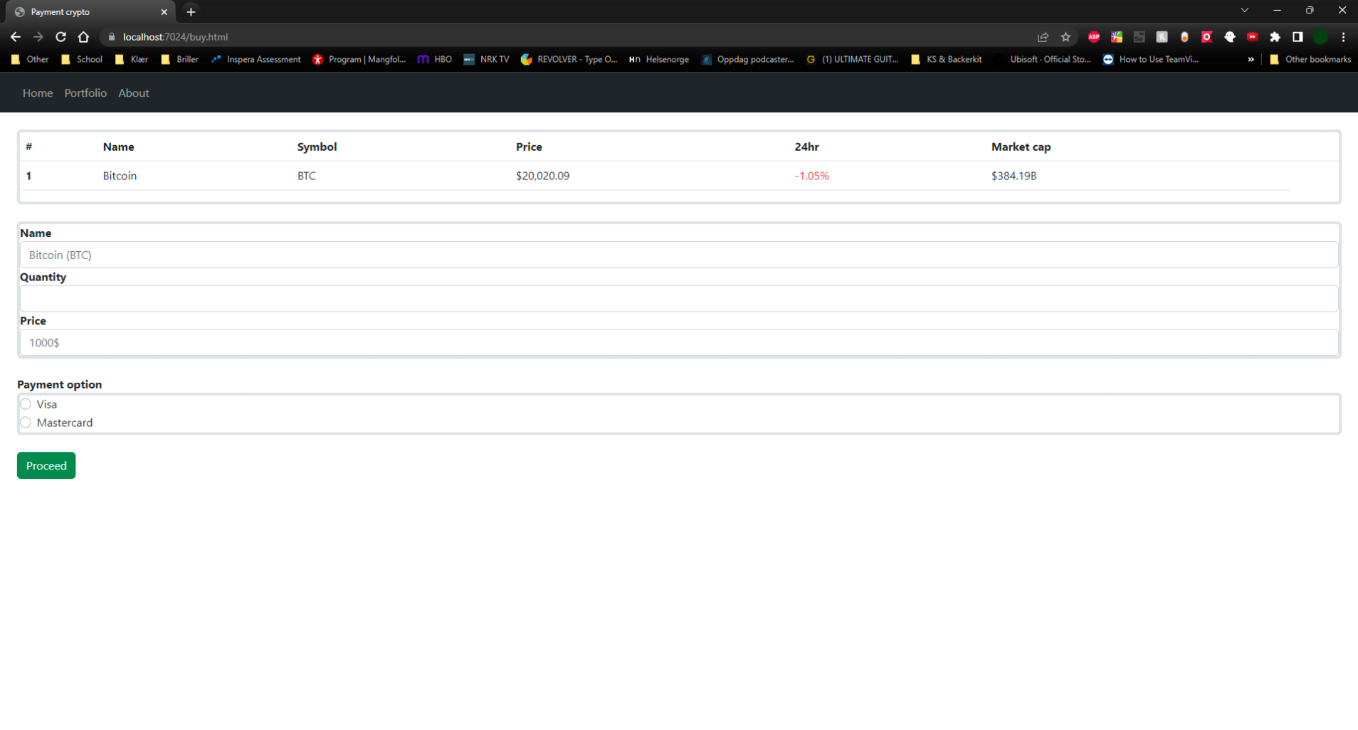


Figur 5 – Kortinndeling basert kjøpsside prototype

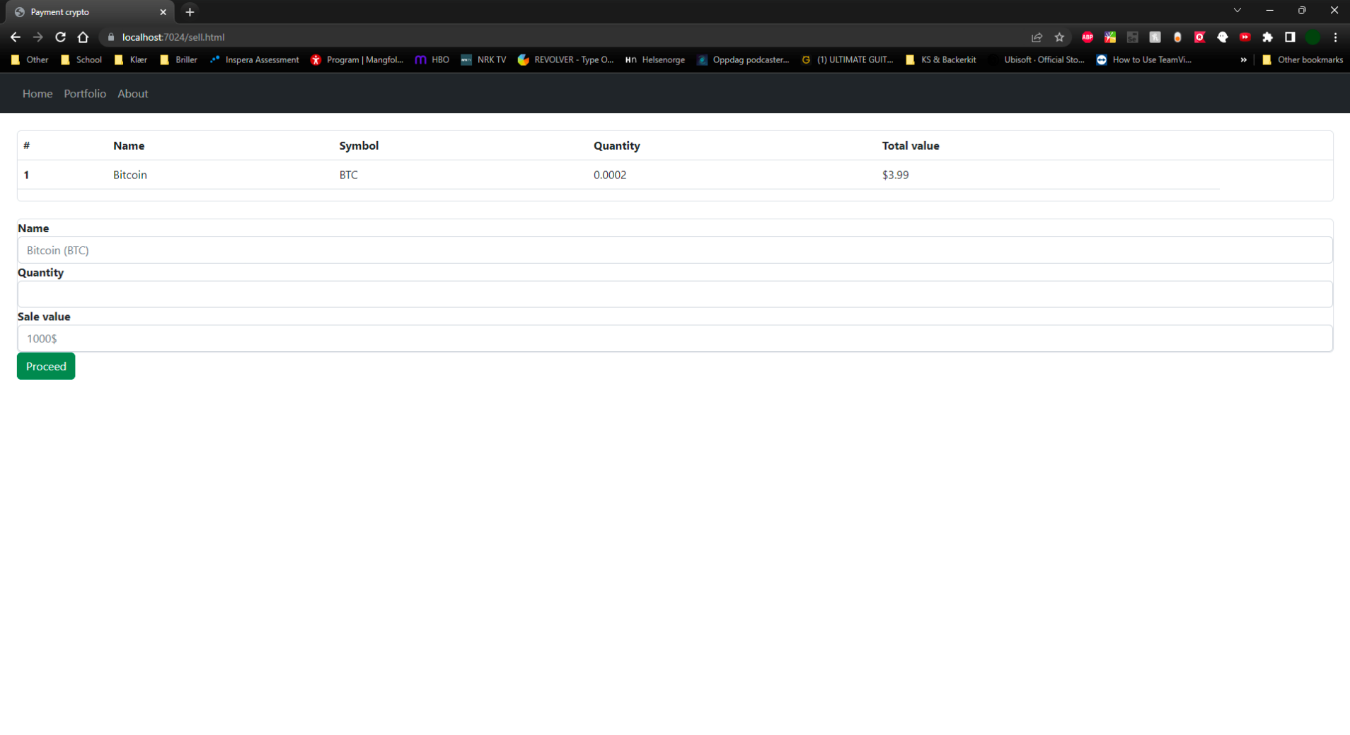


Figur 6 – Kortinndeling basert salgsside prototype

Denne iterasjonen av prototype fokuserer på å sikre hva som må med i løsningen og kategorisere dem i separate kort. Prototypen ble lagd med hensikt å vise hvilke elementer et kjøp og salgs side må ha med i seg og hvordan det vil endres etter hvilke muligheter man gir som for eksempel betaling med VISA mot betaling med faktura. Det er også første prototype der utsende av nettsiden er håndtert av Bootstrap og fungerer i alle skjerm størrelser og formater.



Figur 7 - Kjøpsside prototype



Figur 8 - Kjøpsside prototype

I denne prototypen var fokuset lagt på inkludering av ekstra info i tillegg til kjøp og salg. Så man har bedre oversikt over nåværende pris, endringer osv. Av en kryptovaluta for å gjøre vurderingen av å selge lettere. Uten å måtte gå tilbake til startsiden. Tilsvarende for salg, men med kryptovaluta eid av bruker. For å gjøre det lettere å forstå hvor mye man har å selge, og verdien for å gi bedre perspektiv.

## Besluttet Design:

### Index.html:

Vi beholdte hjemmesidens inndeling i to tabeller der den ene viser tilgjengelige aksjer og deres informasjon, og en tabell som viser hva du har kjøpt. Endringen fra prototypen er for det meste i utsende. Vi la til kortinndelings stiliseringen fra en av kjøpsside og salgsside prototypene og gikk fra tradisjonell CSS til Bootstrap sånn at siden kunne håndtere alle skjermstørrelser og formater. I tillegg la vi til en oppdateringsknapp for kjøpte aksjer sånn at vi har et CRUD element.

Graphical user interface, application

Description automatically generatedFigur 9 - Bilde av hjemmesiden

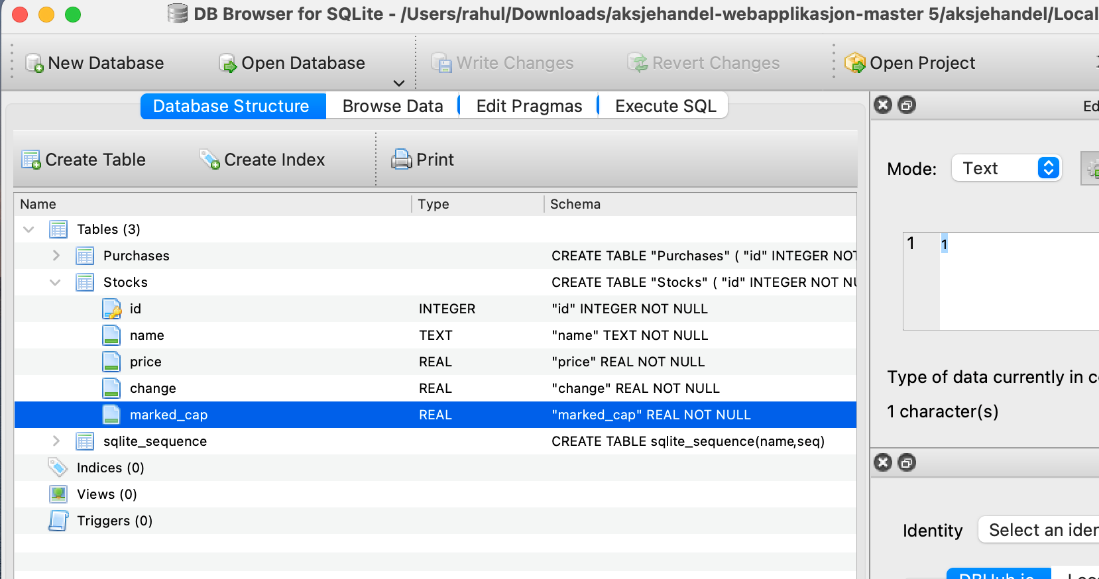
Graphical user interface

Description automatically generated

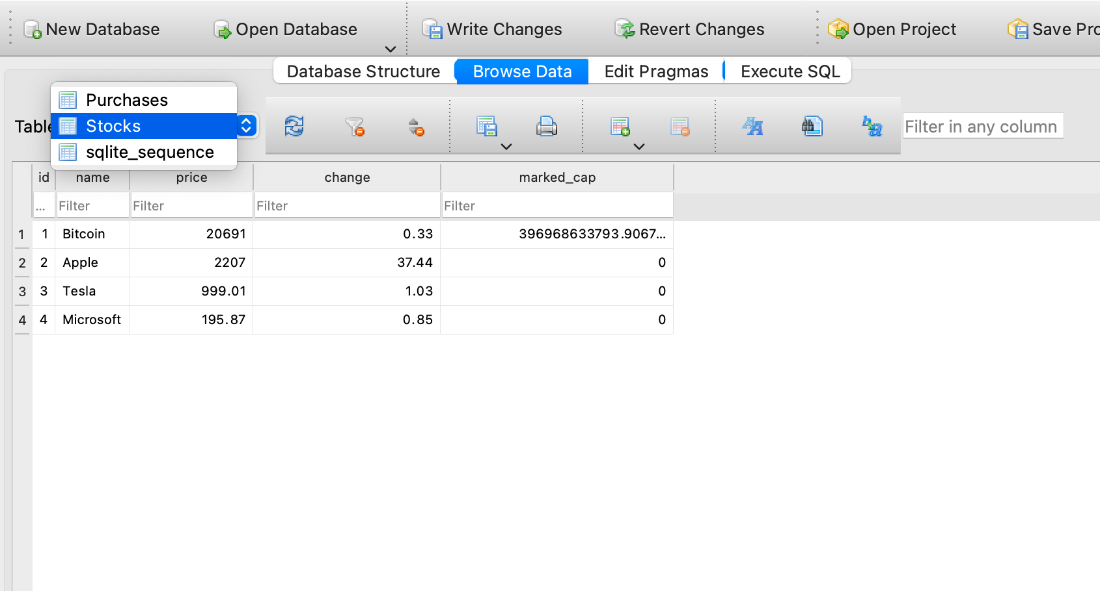
Figur 10 - Bilde av hjemmeside på mindre vertikal skjerm som viser effekt av Bootstrap

### NB :

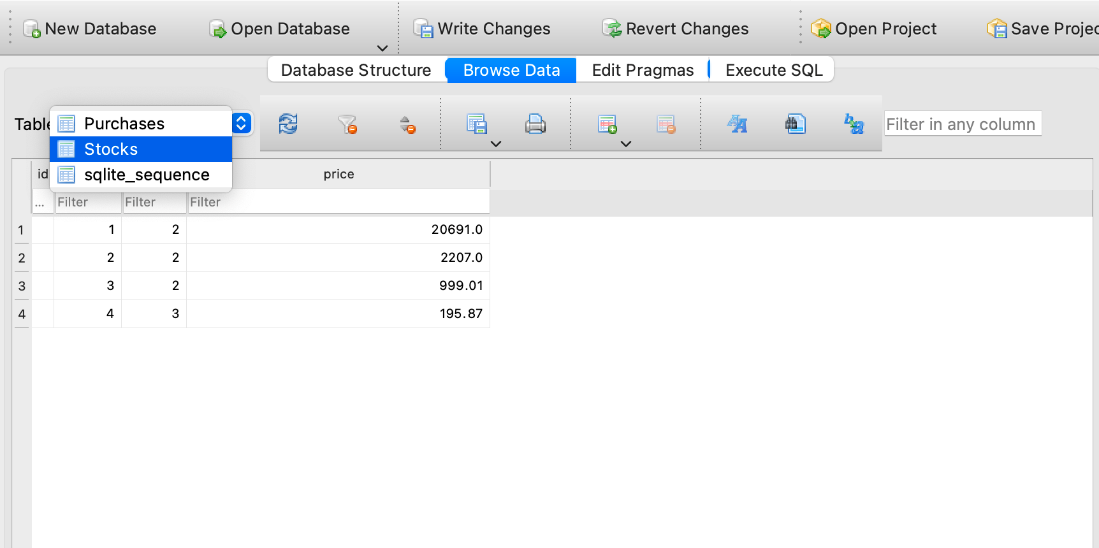
Hvis man skal endre prisene til aksjen, så må man manuelt forandre det i programmet " *DB Browser for SQLite* “.



Figur 11 - Bilde av SQLite 1/3

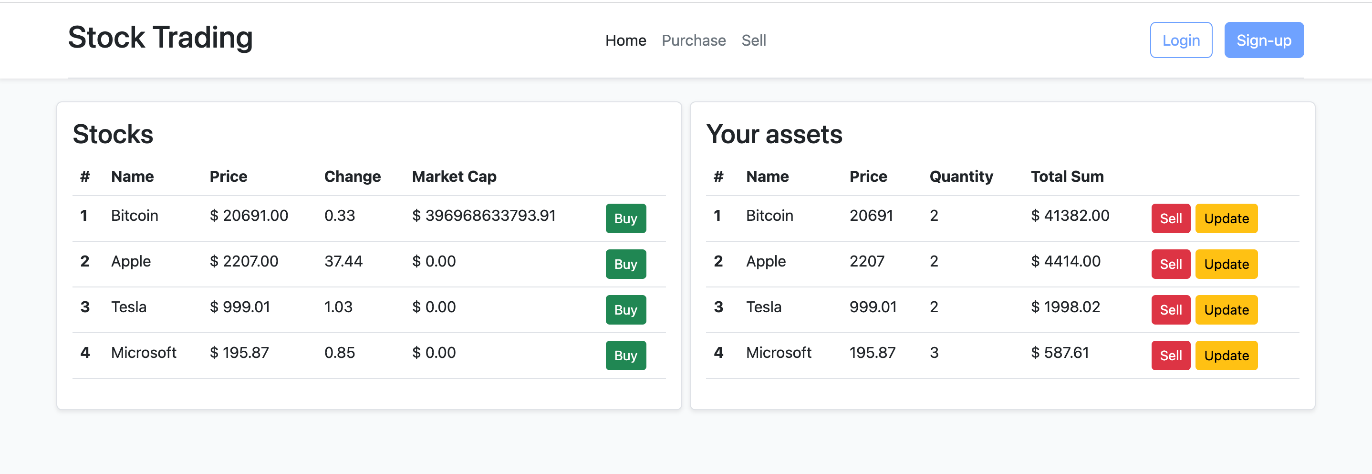


Figur 12 - Bilde av SQLite 2/3

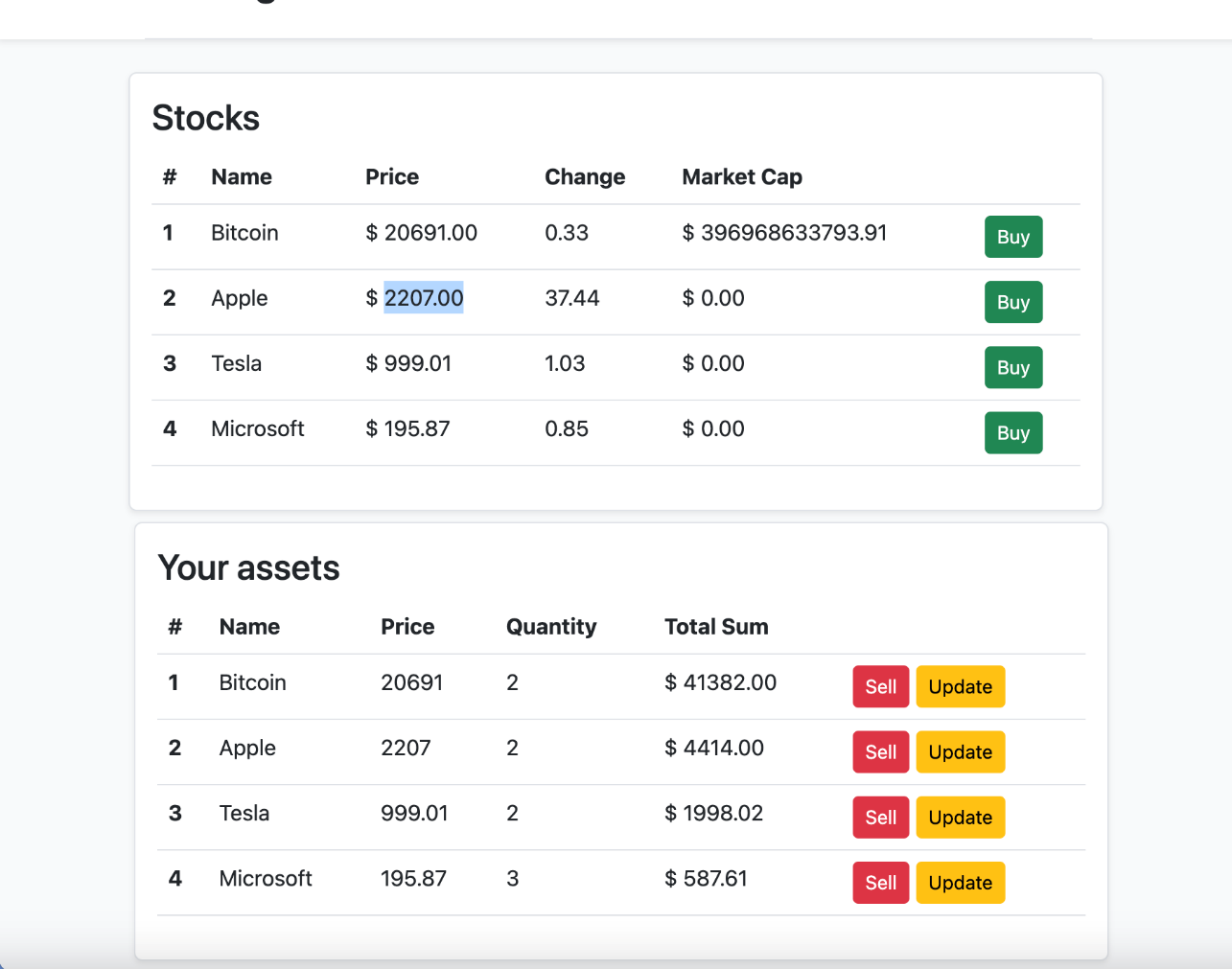


Figur 13 - Bilde av SQLite 3/3

**Resultat:**

3

Figur 14 – Bilde av hjemmeside med oppdatert data ½

4

Figur 15 – Bilde av hjemmeside med oppdatert data 2/2

5

### Kjøp.html

Kjøpssiden er stort sett lik prototypen som er basert på Bootstrap. Endringer som er gjort er sånn at utsende språket er likt de andre sidene. I tillegg fjernet vi betalingsmetoder. Dette ble gjort under beslutning av at det var utenfor en MVPs egenskaper.

Graphical user interface, text, application, email

Description automatically generated

Figur 16 6- bilde av kjøpssiden

Graphical user interface, text, application

Description automatically generated

Figur 177 - bilde av kjøpsside på mindre vertikal skjerm som viser effekt av Bootstrap

### Sell.html

Samme endringer er blitt gjort til salgssiden om kjøpssiden.

Graphical user interface, text, application, Teams

Description automatically generated

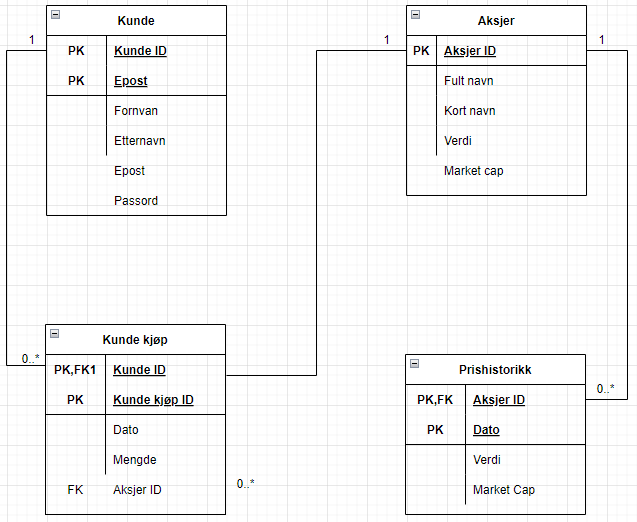
Figur 18 - bilde av salgssiden

Graphical user interface, application

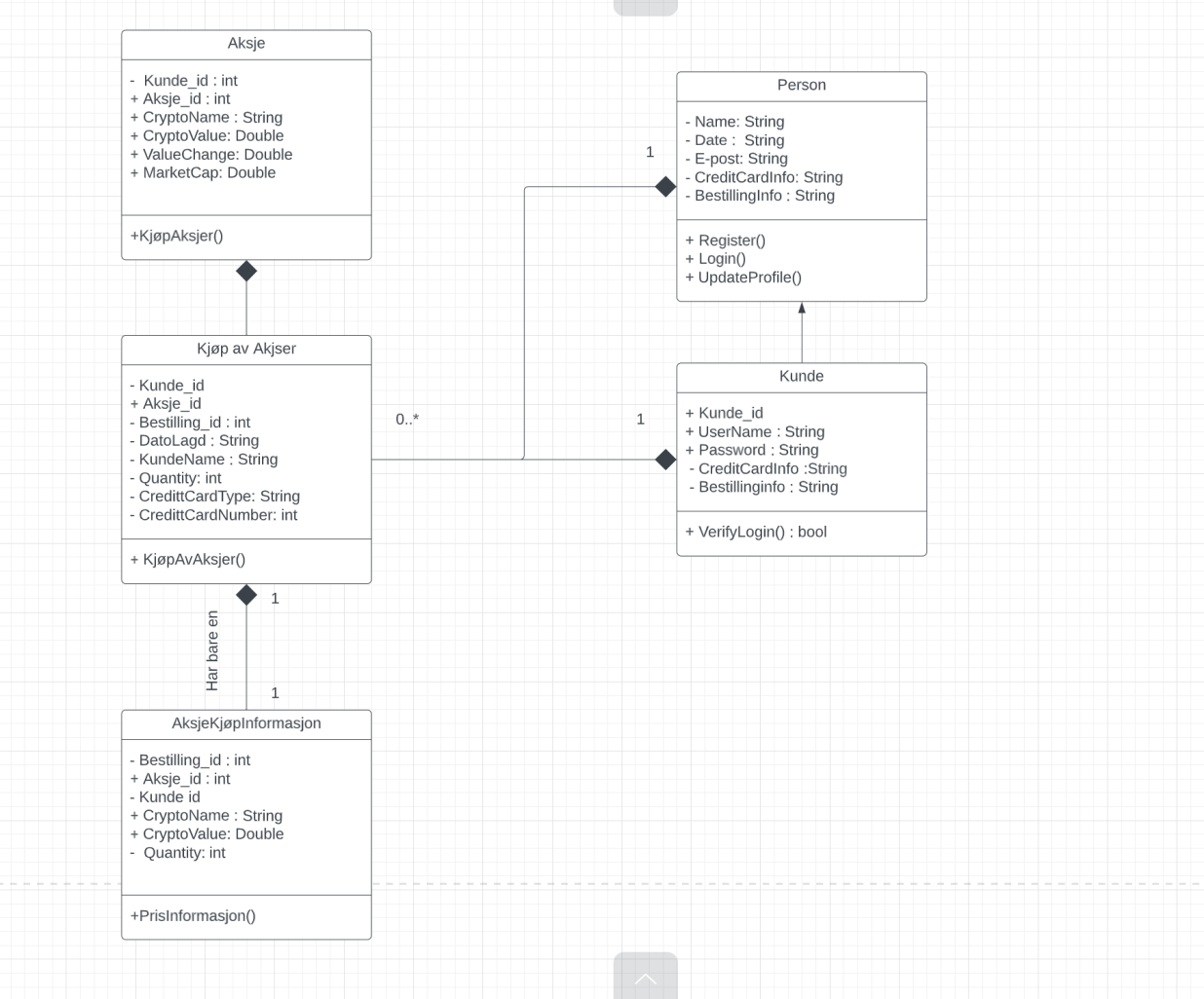
Description automatically generated

Figur 198 - Bilde av salgssiden i mindre vertikal skjerm som viser effekt av Bootstrap

## Diagrammer



Figur 20 9- ER-diagram som beskriver tabeller i databasen

[[1]](#footnote-2)  


Figur 21 10- Klassediagram av databasen

# MVP (Minimal viable product):

Alle knapper inkludert skal føre til et endpoint eller i det minste gi annen log respons.

### Index.html:

* **Navbar:**
  + Home(index.html), purchase(purchase.html) og sell(sell.html).
* **Liste med en type kryptovaluta:**
  + #, name, price, change, market cap og buy(purchase.html).
* **Portfolio med en type kryptovaluta:**
  + #, name, price, quantity, total sum og sell(sell.html)

### Purchase.html:

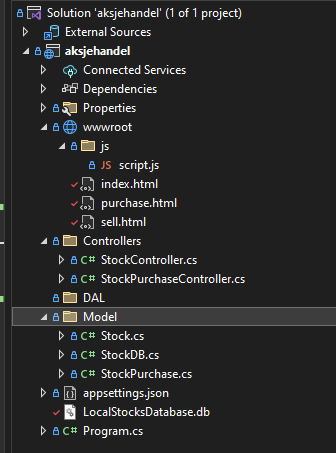
* **Navbar:**
  + Home(index.html), purchase(purchase.html) og sell(sell.html).
* **Liste med en type kryptovaluta:**
  + Type of cryptovaluta, kvantitet og confirm purchase(knapp).

### Sell.html:

* **Navbar:**
  + Home(index.html), purchase(purchase.html) og sell(sell.html).
* **Liste med en type kryptovaluta:**
  + Type of cryptovaluta, kvantitet og confirm sale(knapp).

# Kode beskrivelse:

## Oppsettet:



Figur 22 – Bilde av mappestruktur til løsning

## Controllers:

### - StockPurchaseController.cs:

Løsningen bruker to forskjellige kontrollere, men vi har valgt å bare vise StockPurchaseController.cs. Dette er fordi strukturen på kontrollerne er like, men denne har flere metoder lagt til. I tillegg er StockPurchase det elementet vi bestemte oss for å gjennomføre CRUD i.

En kontroller blir opprettet ved definere en “route” til ASP NET Core som forteller applikasjonen hvilken URL man bruker for å lese og oppdatere ressurser til API’en. Her definerer man HTTP metoder som tilsvarer forskjellige handlinger man kan gjøre mot en ressurs. For eksempel, hvis man ønsker å hente data, bruker man en GET metode, og tilsvarende en POST metode for å sende inn data.

Kontroller kobler seg til databasen ved å ved å initialisere kontroller-klassen med en database kontekst som kommer fra “Models” mappen. Denne modellen har enkle metoder som gjør det mulig å hente og lagre data fra databasen.

Kontroller har fire egenskaper og disse kombinert gjør at vi oppnår CRUD. Alle egenskapene bruker httpget eller httppost til å hente eller sende data etter hvilke egenskapen gjør.

Den første er GetAll() og gjør at vi kan vise alle aksjekjøpene. Den gjør dette ved å lage en variable av typen “List<StockPurchase>” som kaller database modelen’s “toList” metode for å hente kjøpene fra databasen. Deretter blir disse variablene returnert. ASP Net Core henter, formatterer og returner alt dette for oss. Den andre kontrolleren for “Stocks” har kun en lik “GetAll” metode som returnerer alle mulige aksjer.

For å oppnå “Create” av CRUD og kjøpet av akjse har vi lagd Save([FromBody] StockPurchase purchase). Denne lager et kjøp ved ta POST forespørselens innhold som input fra HTTP body’en og legger denne inn i databasemodellen’s “purchase” liste og deretter kaller “SaveChanges” for å lagre den nye daten. ASP Net sjekker ikke om innholdet er i riktig format som gjør at det er viktig å opprette input validering her også for å ha integritet i dataen.

Update egenskapen er noe som ikke gir fullstendig mening med kjøpt aksje, men er der for å oppnå CRUD. Update(int id, [FromBody] StockPurchase newPurchase) brukes til å endre en kjøpt aksje og gjør dette på en lignende måte som “Create” metoden ved at den tar inn en HTTP body med et oppdatert aksjekjøp. I tillegg til dette så henter den aksje kjøp ID’en fra URLen som at den vet hvilken ressurs som skal endres. Denne metoden prøver først å finne et aksjekjøp som har samme ID, hvis den er funnet så blir klassen oppdatert med ny data og lagret i databasen. Hvis aksjekjøpet ikke blir funnet, så returnerer APIen “false”. Dette forsikrer at man ikke kan oppdatere et kjøp som ikke finnes.

Til slutt har vi lagt til slett egenskapen. Denne egenskapen blir brukt til å selge en kjøpt aksje. Delete(int id) sletter en kjøpt aksje. Denne metoden tar inn et aksjekjøp ID, sjekker om kjøpet finnes og sletter det fra databasen. Her er det også validering som sjekker om kjøpet først finnes.

Text

Description automatically generated

Figur 23 - Bilde av StockPurchaseController.cs 1/2

A screenshot of a computer

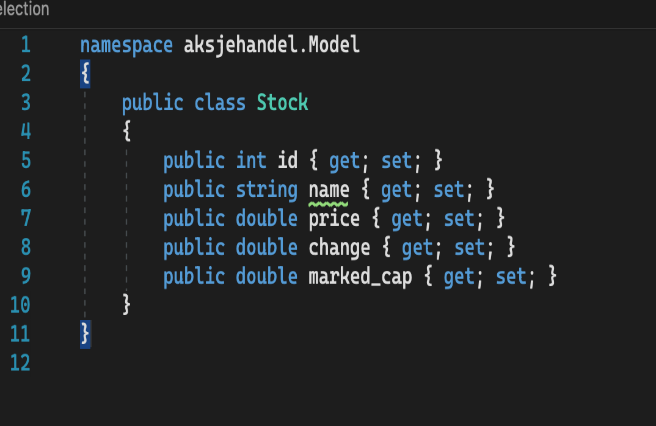
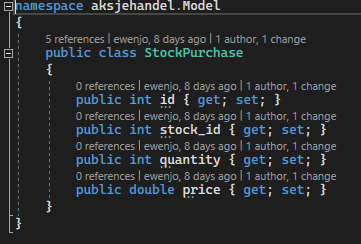
Description automatically generated with medium confidence

Figur 24 - Bilde av StockPurchaseController.cs 2/2

## Model:

### Stock.cs og StockPurchase.cs:

Applikasjonen består av tre modeller. To modeller for database tabeller og en for database kontekst. Stock og StockPurchase modellene beskriver tabellene og ressursene til APIen. Her defineres hvilke attributer/kolloner som finnes og hvilken data type de har. Begge modellene har GET og SET metoder som gjør det mulig å lese og oppdatere dataen.

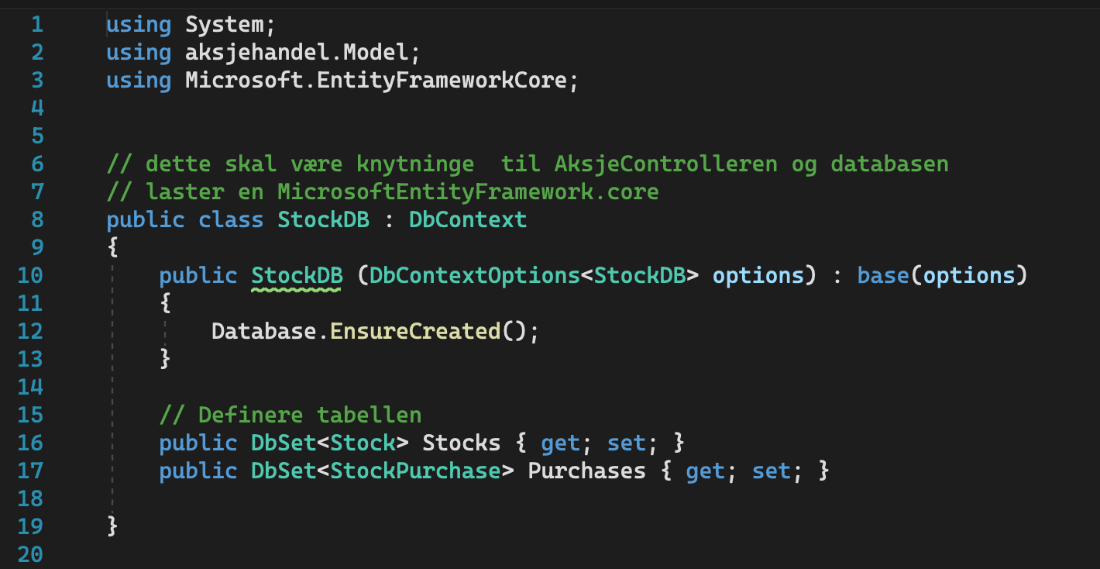


Figur 25 11- Bilde av StockPurchase.cs sin kode Figur 26 12- Bilde av Stock.cs sin kode

### StockDB.cs:

StockDB modellen er litt annerledes. Når modellen blir opprettet, så brukes “EnsureCreated” for å sjekken om database filen finnes, hvis ikke så blir den laget. Denne modellen har også to "DbSet” attributter som tilsvarer tabellene til databasen. Her brukes Stock og StockPurchase modellene til å definere typen på tabell kolonnene.

Den lokale databasen kommer ferdig og inneholder aksjedata. Vi har ikke benyttet “seeding” enda.



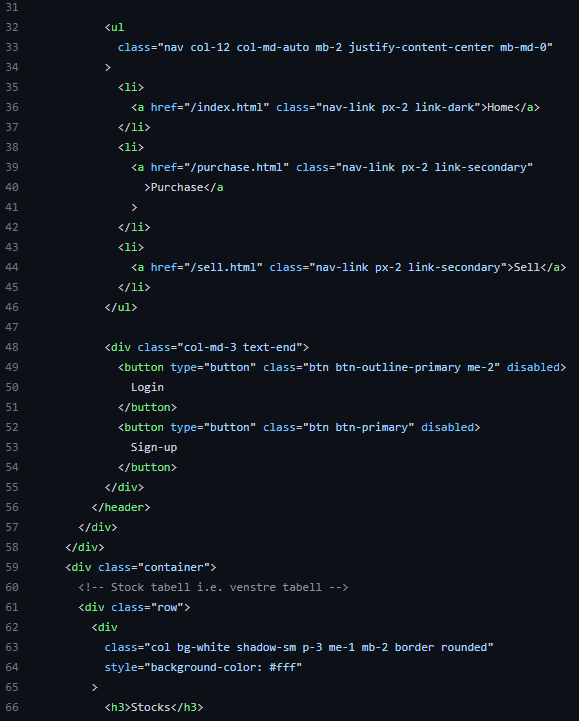
Figur 27 – Bilde av stockDB.cs

## wwwroot:

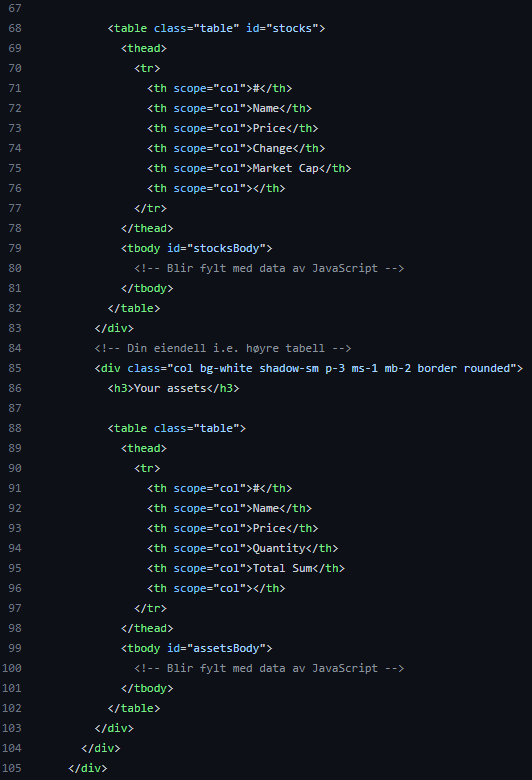
### index.html:



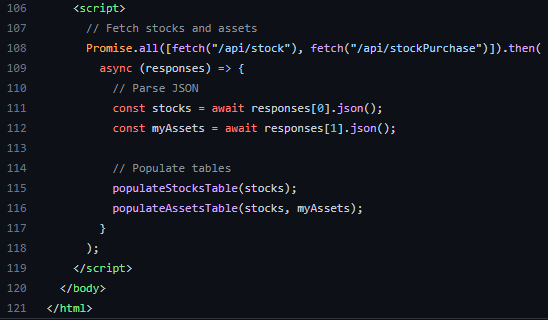
Figur 28 - Bilde av index.html del 1/4



Figur 29 13- bilde av index.html del 2/4



Figur 30 14- Bilde av index.html del 3/4



Figur 31 15- Bilde av index.html del 4/4

### purchase.html:



Figur 32 16- Bilde av purchase.html del 1/6



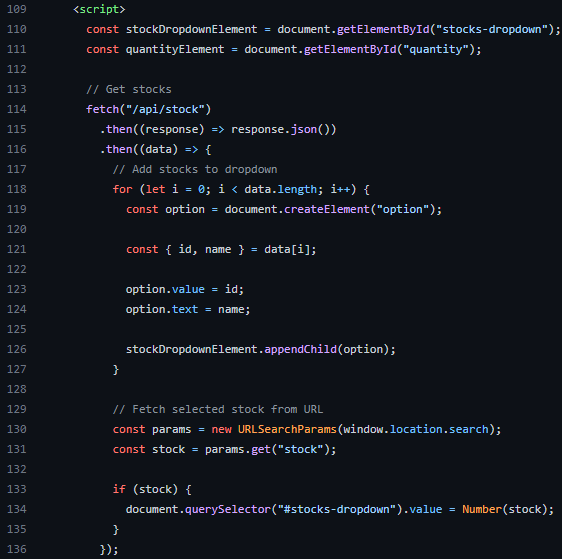
Figur 33 17- Bilde av purchase.html del 2/6



Figur 34 18- Bilde av purchase.html del 3/6



Figur 33 - Bilde av purchase.html del 4/619

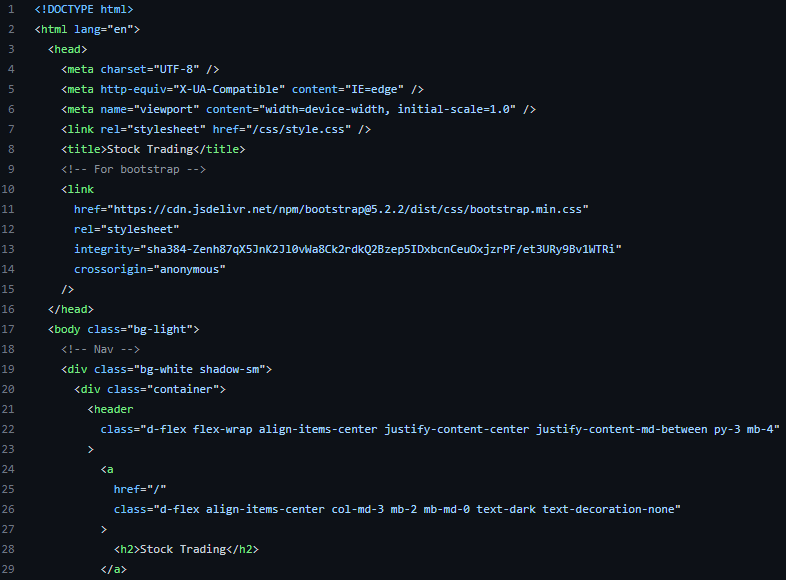


Figur 35 20- Bilde av purchase.html del 5/6



Figur 36 21- Bilde av purchase.html del 6/6

### sell.html:



Figur 37 22- Bilde av sell.html del 1/5



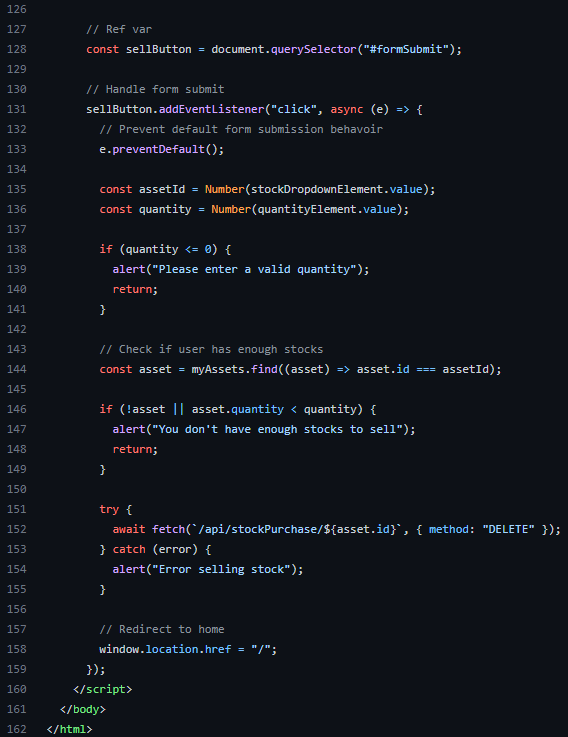
Figur 38 23- Bilde av sell.html del 2/5



Figur 39 24- Bilde av sell.html del 3/5

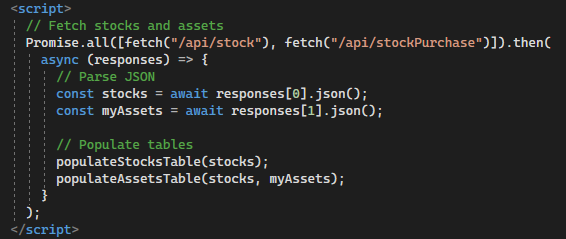


Figur 40 25- Bilde av sell.html del 4/5



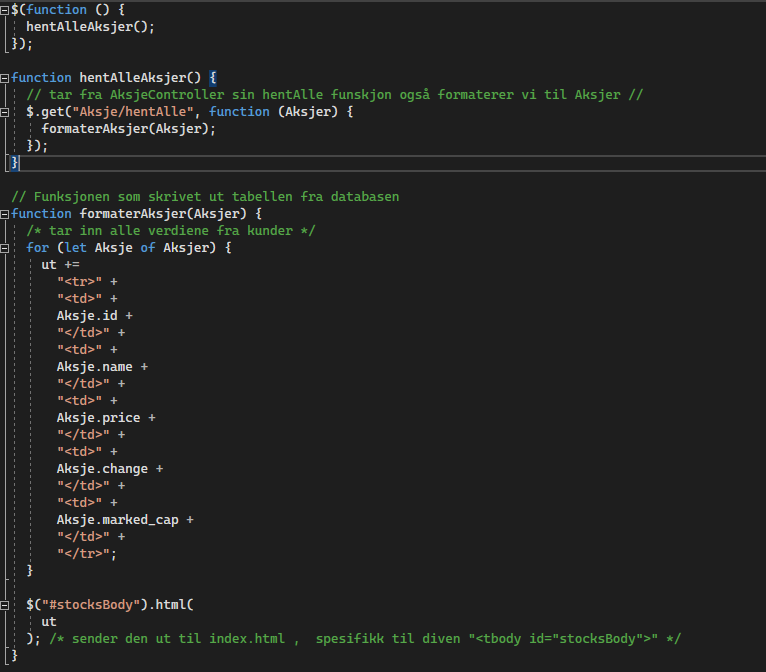
Figur 41 26- Bilde av sell.html del 5/5

### Script.js:



Figur 42 - Bilde av index.html sin JavaScript

I index.html har vi et script som henter aksjer data og legger det inn i tabell.



Figur 43 - Bilde av script som blir brukt av index.html til å konstruere aksjer tabellene

Tabell opprettelse håndteres i en annen JavaScript-fil. Text

Description automatically generated

Figur 44 - Bilde av purchase.html script som legger inn aksje dropdown verdier

I purchase.html har et script som legger inn aksjene i dropdown meny som lar deg velge en aksje du vil kjøpe.

Text

Description automatically generated

Figur 45 - Bilde av purchase.html script som gjør kjøp

Denne script blir brukt til å lagre kjøpet du gjør på purchase.html.

Text

Description automatically generated

Figur 46 - Bilde av sell.html script for henting av eid aksje

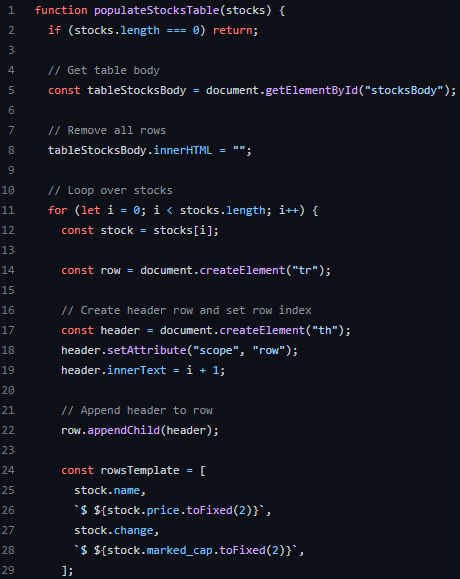
Script bruk i sell.html til å hente aksjene du eier.

Text

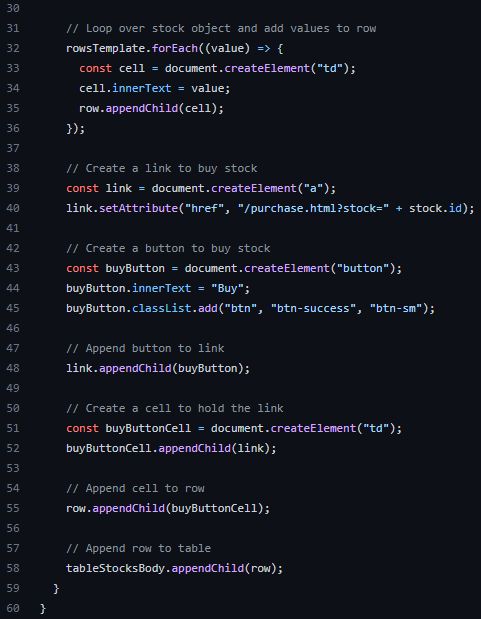
Description automatically generated

Figur 47 - Bilde av sell.html script som gjør salg

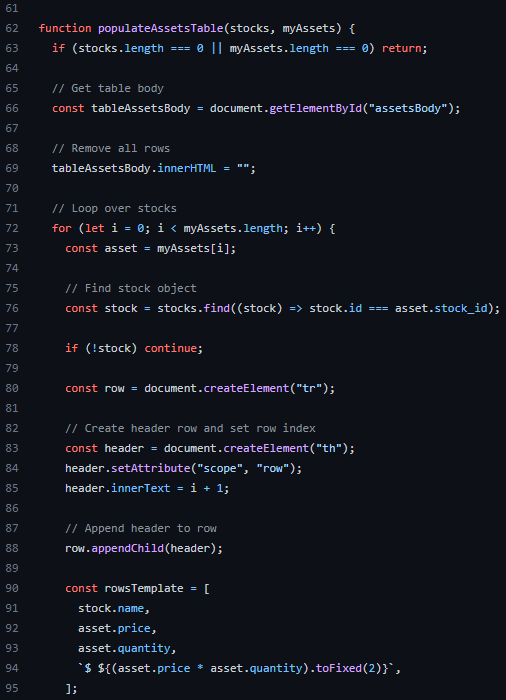
Script til sell.html som gjør salg av aksje.



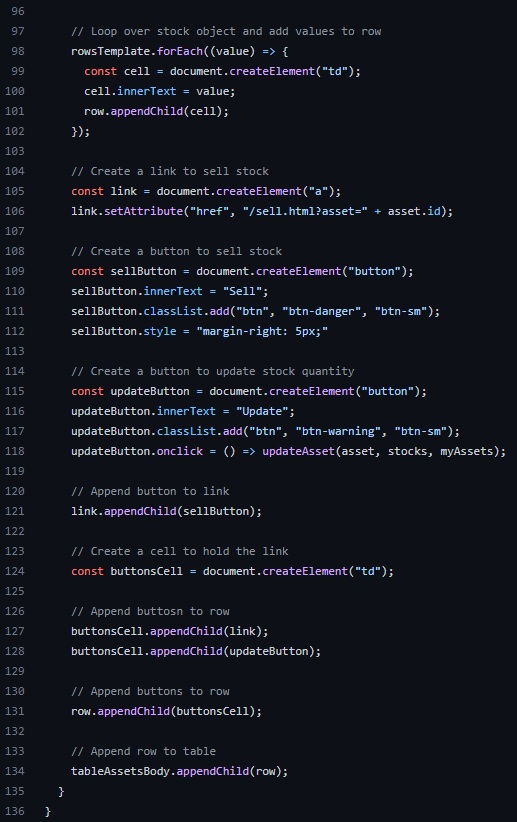
Figur 48 – Bilde av script.js 1/5



Figur 49 – Bilde av script.js 2/5



Figur 50 – Bilde av script.js 3/5



Figur 51 – Bilde av script.js 4/5



Figur 52 – Bilde av script.js 5/5

# Konklusjon:

For å konkludere har vi fullført Aksjehandel siden. Det har blitt laget tre sider med navn index.html (hovedsiden), purchase.html (Kjøpe prosessen) og sell.html (selge prosessen). På hovedsiden kan du se tilgjengelige aksjer og hvilke aksjer du eier. Aksjene du eier kan du selge og endre på, og dette gjør at det oppnår CRUD i samspill med kjøp av aksjer. Kjøp siden gir deg mulighet til å kjøpe tilgjengelige aksjer. Den kan nåes ved å trykke kjøp knappen på en av aksjene på hovedsiden eller bruke navigasjonsbaren. Når man har kjøpt en aksje kan man gjøre endringer på den på hovedsiden eller selge den ved å trykke på selg knappen eller bruke navigasjonsbaren.

# Referanser:

## **Youtube:**

* Code Prime. (2022, 2. feb). *How To Show The Live Price of CryptoCurrency on website using HTML CSS, and Javascript [Video]. Youtube.*

[*https://www.youtube.com/watch?v=VDLpDSZ88MM*](https://www.youtube.com/watch?v=VDLpDSZ88MM)

* freeCodeCamp.org. (2021, 7. des). *Bootstrap CSS Framework - Full Course for Beginners [Video]. Youtube.*

[*https://www.youtube.com/watch?v=-qfEOE4vtxE*](https://www.youtube.com/watch?v=-qfEOE4vtxE)

* freeCodeCamp.org. (2018, 10. des). *Learn JavaScript - Full Course for Beginners* [Video]. *Youtube.*

[*https://www.youtube.com/watch?v=PkZNo7MFNFg*](https://www.youtube.com/watch?v=PkZNo7MFNFg)

## **Akjsenettsider:**

* Coinbase. (2022). *Jump start your crypto portfolio* [Website].

[*https://www.coinbase.com/*](https://www.coinbase.com/)

* Tradingview. (2022). *Look first Then leap [Website].*

[*https://www.tradingview.com/*](https://www.tradingview.com/)

* *Investor.d. (2022). Hovedindeksen på Oslo [Website].*

[*https://investor.dn.no/#!/Indeks/X3/OSEBX/Hovedindeksenp%C3%A5OsloB%C3%B8rs*](https://investor.dn.no/#!/Indeks/X3/OSEBX/Hovedindeksenp%C3%A5OsloB%C3%B8rs)

## **Api:**

* *Coingecko. (2022). Explore the API [Website]*

[*https://www.coingecko.com/en/api/documentation*](https://www.coingecko.com/en/api/documentation)

[*https://api.coingecko.com/api/v3/simple/price?ids=bitcoin%2Ctesla%2Capple%2C%20microsoft&vs\_currencies=usd&include\_market\_cap=true&include\_24hr\_change=true*](https://api.coingecko.com/api/v3/simple/price?ids=bitcoin%2Ctesla%2Capple%2C%20microsoft&vs_currencies=usd&include_market_cap=true&include_24hr_change=true)

1. [↑](#footnote-ref-2)