**Database modellen**

I dette prosjektet har vi benyttet Sqlite databasen og Entity Framework core som ORM (object-relational mapper) for tilgang til databasen fra .net core web applikasjonen. Database modellen som er blitt konstruert til denne oppgaven består av entitetene Users, Stocks, Trades, SearchResults og StockQuotes. Videre eksisterer det mange til mange forhold mellom Users og Stocks. En bruker kan eie flere aksjer og ha flere aksjer i sin favorittliste. Videre kan en aksje være i favorittlisten til flere brukere og være kjøpt av flere brukere. Videre er det et mange til mange forhold mellom SearchResults og Stocks ettersom flere aksjer kan være i et søkeresultat og en aksje kan være del av ulike søkeresultat.

For å representere disse mange til mange forholdene, opprettes hjelpetabeller (join tables). I utgangspunktet vil entity framework gjøre dette automatisk dersom et mange til mange forhold er definert implisitt i DbContext programmet. EF vil da sette sammen navn på tabell og attributter ut fra de to entitetene som inngår i forholdet. I dette tilfelle er alle navnene til hjelpetabellene definert eksplisitt: FavoriteLists, StockOccurances og StockOwnerships. Merk at StockOwnerships er en tabell som inneholder flere attributter en bare primær og fremmednøklene (SpentValue og StockCounter). Derfor er denne tabellen definert som en egen entitet i TradingContext. For at EF skal opprette tabellen med både Symbol (aksje id) og UsersId som både primær og fremmednøkkel er det viktig at StocksId og UsersId er definert som attributter i tillegg til navigation property for både Stocks og Users tabellen.

**Database migrasjoner og seeding**

Med Entity Framework finnes flere metoder for å bygge databasen ut fra DbContext. I dette tilfelle har vi benyttet migrations funksjonaliteten i stedet for å benytte EnsureCreated eller EnsureDeleted. Ved hjelp av migrations kan eksisterende data i databasen bevares ved endringer av database modellen. Videre er det kun lagt til et brukerobjekt som skal legges til i databasen når den opprettes (seeding). Hver gang det foretas endringer i database modellen (f.eks legge til en attributt) kan man lage en ny migrasjon med «Add-Migration migrationName» og deretter bygge databasen på nytt baserende på den siste migrasjonene med «Update-Database»

**Worker tjeneste for rydding av databasen**

Lagring av søkedata kan føre til at mye data i form av aksje objekter og søkeresultat objekter er lagret i databasen uten at det blir aktivt brukt av brukere. I forbindelse med dette er det blitt lagt til en metode i TradingRepository: CleanTradingSchemaAsync(). Denne metoden fjerner søkeresultater som ble lagret for over 24 timer siden. Videre fjernes alle Stocks objekter og tilhørende StockQuotes objekter som ikke har noen relasjon til en bruker (portfolio eller favorittliste) eller et søkeresultat.

For at denne metoden kjøres en gang daglig, er det implementert en hosted service, «TradingSchemaWorker», som starter når applikasjonen startes og kjører i bakgrunnen som en uavhengig tjeneste. TradingSchemaWorker klassen er basert på Worker malen i dotnet. Videre defineres det et Timer objekt i StartAsync metoden, som starter en syklus der CleanTradingSchemaAsync() kalles når klokken til maskinen som kjører programmet er 00:00:00. Merk at «CleanTradinSchemaAsync» metoden kjøres en gang når ved oppstart av applikasjonen.