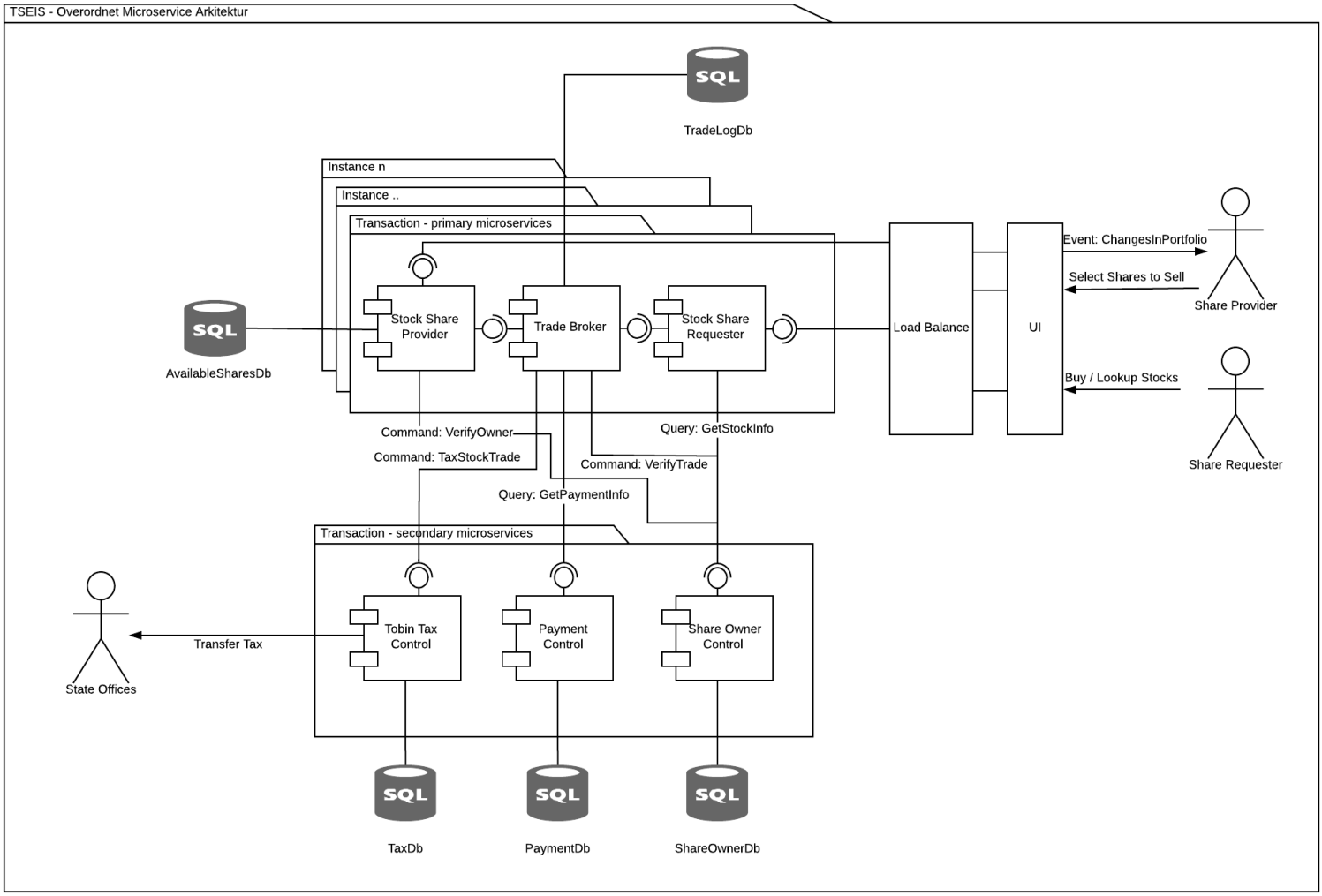
23/02 - 2018

The Stock Exchange Interaction System

F18 ITONK Projektgruppe 7
Aleksander Konnerup (201505486)
Simon Schlüter Pleidrup (201507214)
Casper Stenner Johannsen (201508656)

Hvilke Microservices?

Nedenfor ses et diagram over TSEIS's overordnede microservice arkitektur, det ses her at microservicene er inddelt efter primære og sekundære. De primære repræsenterer de microservices der benyttes til at lave selve overførslen af aktier, herunder menes de tre microservices: Stock Share Provider, Trade Broker og Stock Share Requester. Ved de sekundære microservices, menes der services der ikke direkte er en del af transaktionen, men benyttes til at modtage eller videregive nødvendig information. Disse er: Payment Control, Share Owner Control og Tobin Tax Control. På nedenstående tegning ses desuden at hver microservice udstiller et API, der ved brug af http-protokollen tillader en række forskellige operationer, disse vil blive beskrevet nærmere i følgende afsnit.



Figur 1: Overordnet Microservice Arkitektur for TSEIS

Som nævnt udstiller hver microservice en række API og derved også en række endpoints, følgende viser en kort beskrivelse af hver microservice og dens endpoints:

Tobin Tax Control:

Per lov fra staten skal der for hver gennemført transaktion tilføjes en beskatning på 1% af det samlede transaktionsbeløb, den såkaldte Tobin Tax. Når en transaktion gennemføres af de primære microservices, sendes transaktionsbeløbet til microservicen, denne udregner Tobin Tax og sender beløbet til staten.

Post:

- PayTobinTax(int transactionValue, guid transactionId)

Payment Control:

For at brugerne af systemet let og hurtigt kan købe/sælge aktier, er det nødvendigt at have en microservice, der tillader at gemme betalingsdata for brugerne. Det er herfra muligt at hente betalingsinformation op for den enkelte bruger.

Get:

- GetPaymentInfo(guid paymentId)

Share Owner Control:

Indeholder en database med beskrivelse af hvem der ejer hvilke aktier, Share Owner Control benyttes til at hente information op om de enkelte aktier, såfremt en bruger ønsker at sælge/købe eller se sine aktier. Ligeledes opdateres databasen for hver gennemført transaktion, derved er Share Owner Control altid ajourført.

Post:

- VerifyShareOwnership(string stockId, guid userId, int shareAmount)
- UpdateShareOwners(guid requester, guid provider, int shareAmount, string stockId)

Get:

- GetStockInfo(string stockName)

Trade Broker:

Trade Broker microservicen fungerer som formidleren, mellem Stock Share Provideren og Stock Share Requesteren. Når en bruger ønsker at købe en eller flere aktier, bliver denne request sendt til Trade brokeren, derfra hiver trade brokeren fat i Stock Share provideren for at høre om aktierne er til salg. Efter transaktionen tager trade brokeren fat i de sekundære microservices og videregiver de nødvendige informationer.

Post:

- InitiateTrade(string stockName, guid requester)

Stock Share Provider:

Stock Share Provideren har til formål at tjekke tilgængeligheden for de enkelte aktier. Såfremt en bruger ønsker at sælge nogle af sine aktier, kan han markere dem som "til salg", denne information bliver gemt af Stock Share Provideren, som senere bruger disse til at informere brokeren om aktierne kan købes.

Post:

- MarkStocksForSale(string stockId, int sharedAmount)

Stock Share Requester:

Når en bruger ønsker at købe eller se aktier, sender Stock Share Requesteren et Request til trade brokeren eller Share Owner Control. Denne microservice har til ansvar at videreformidler brugerens request efter dette er gjort sendes den nødvendige information til andre microservices og resten af forløbet udføres. Efter operationen er udført valideres den ved trade brokeren, derefter sendes resultatet gennem Stock Share Requesteren og videre ud til brugeren.

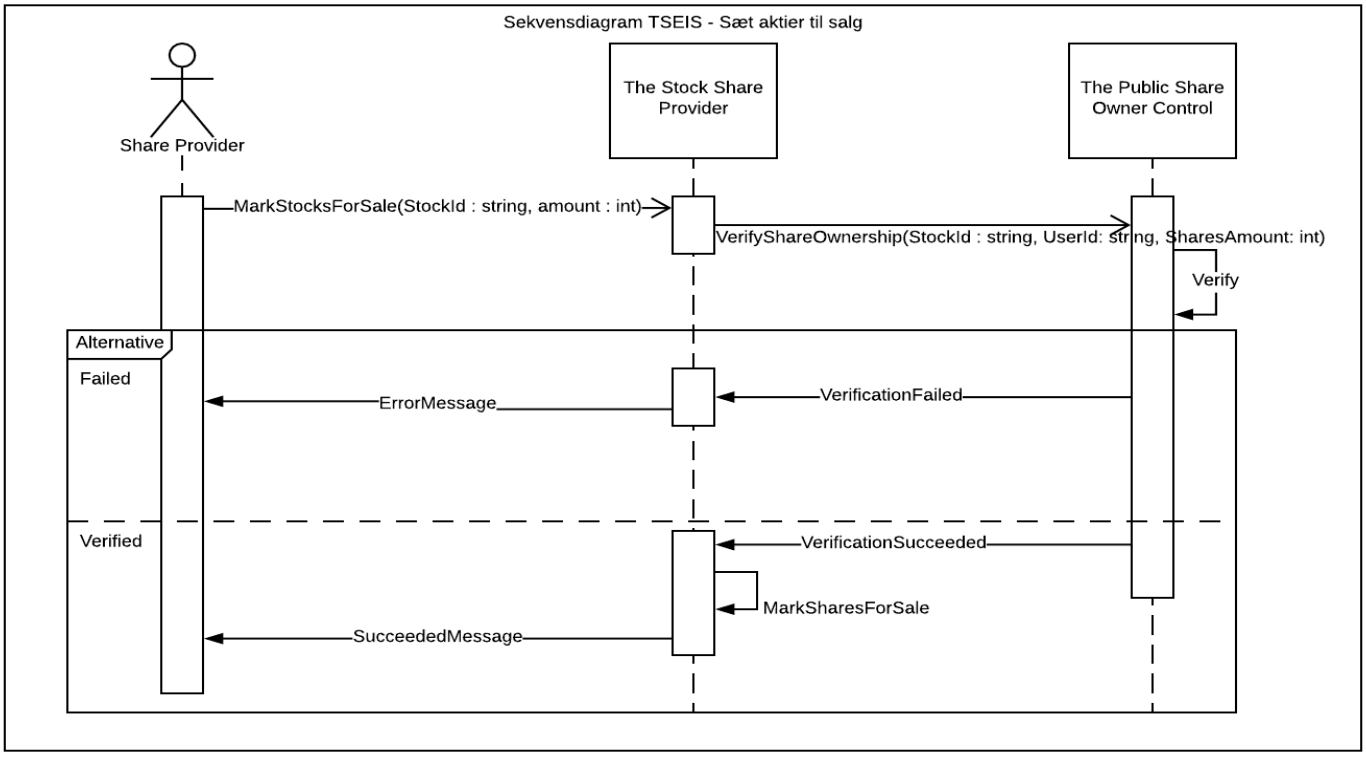
Endpoints:

- Udstiller kun endpoints ud til brugeren.

Vigtigste forløb ift. Krav

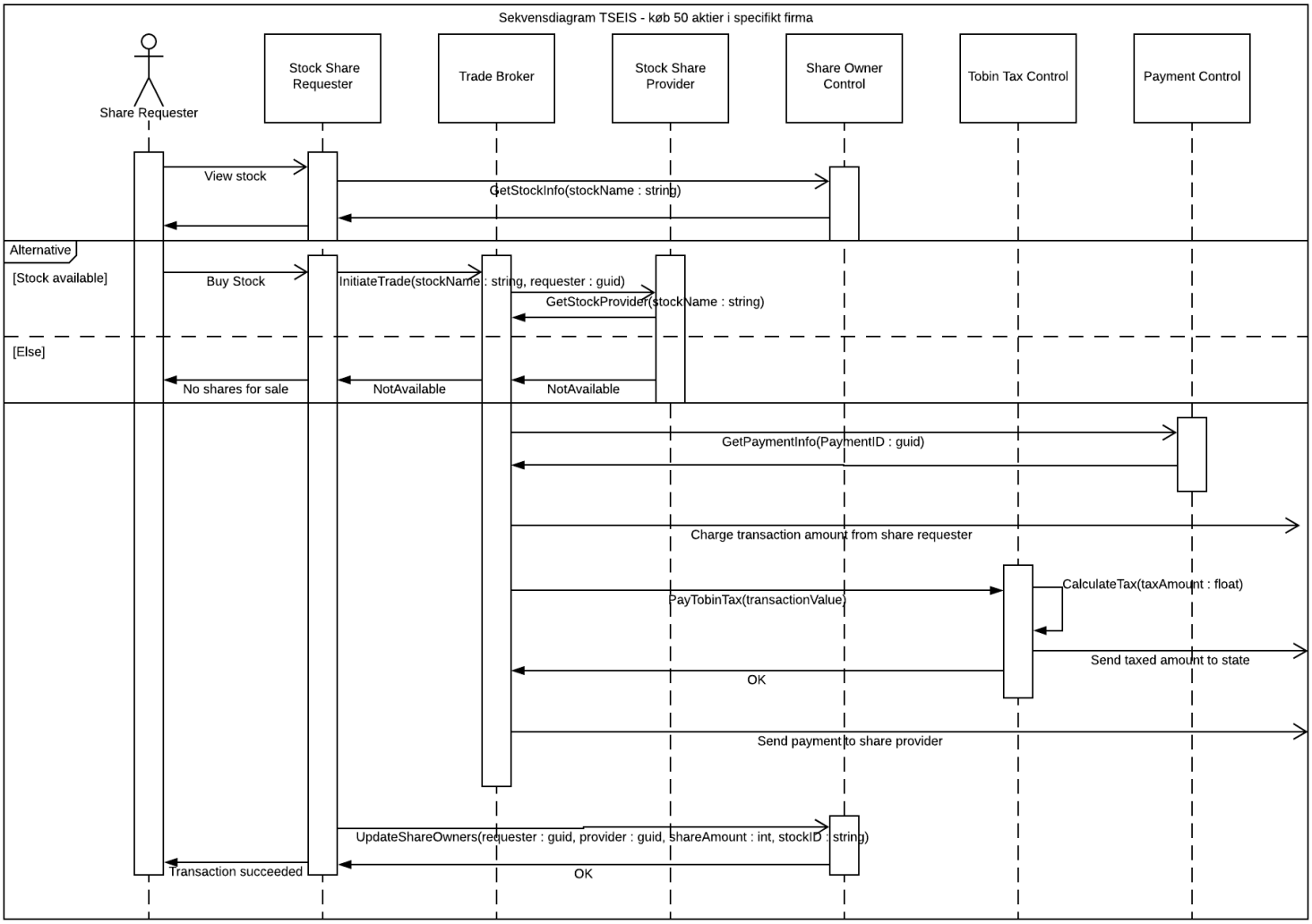
I følgende afsnit ses to sekvensdiagrammer, der demonstrerer hovedforløbene i vores system.

I nedenstående sekvensdiagram antages det, at Shared Provideren allerede har hentet information om hvilke aktier brugeren er i besiddelse af.



Figur 2: Sekvensdiagram Sæt aktier til salg

Følgende sekvensdiagram viser scenariet hvor en bruger køber 50 aktier fra et vilkårligt firma.



Figur 3: Sekvensdiagram køb aktier

Refleksion og yderligere krav

Under udarbejdet af arkitekturen for TSEIS, gjorde vi os en lang række overvejelser. Herunder specielt vedrørende præcis hvilke data, der er nødvendig for de enkelte microservice og hvilke data man som bruger ønsker at have adgang til. Dette kunne fx være overblik over tidligere transaktioner, det blev her besluttet at disse skulle gemmes i Trade Brokerens Database. Yderligere gjorde vi os en række overvejelser vedrørende Error-logging, der blev her overvejet om det gav mening at lave en microservice til dette. Det blev dog i sidste ende besluttet at et allerede eksisterende library som fx Kibana, allerede tilbød den funktionalitet vi ønsker og kræver ikke yderligere ændringen til arkitekturen.

I forhold til yderligere krav er det mest markante nok vores ”Payment Control” microservice. Denne er tilføjet med basis i et selvopstillet krav om at det som bruger skal være let at tilgå sine betalingsoplysninger, for at gøre det hurtigere og nemmere at købe eller sælge aktier.

Slutteligt har vi valgt at lave ID’er, for bruger og Payment Info, som Guid. Dette er gjort for at gøre det nemmere at parse ID’erne rundt i systemet, da disse skal benyttes af flere forskellige microservices.