

Dato for aflevering: 7.juni 2020
Underviser: Jesper Rosholm Tørresø
Fag: ITONK

TSEIS

Gruppe 19

Valeria Polukhina Wellejus 201507658

Indhold

Indledning.....	2
Krav til systemet	2
Mountain Goat format user stories fra projektformuleringen	2
Use cases	2
Use case 1: Bruger sætter en aktie til salg	2
Use case 2: Bruger køber en aktie	3
Skitser til UI.....	3
Arkitektur.....	5
Oversigt over services.....	5
Data view	6
Implementation view	7
Deployment view.....	8
Transparens i applikationen	10
Applikationen opfylder.....	10
Applikationen opfylder ikke.....	10
Resultater	10
Forbedringer	11
Referencer	12

Indledning

I denne rapport vil der beskrives en simpel løsning på arkitekturen til 'The Stock Exchange Interaction System'. Systemet består af flere microservices, som ikke er tilknyttet den samme data storage. Microservices er deployed på GKE F19ITONK clusteret og applikationen kan tilgås via external endpoints. Besvarelsen er udarbejdet af gruppe 19, som består af Valeria P. Wellejus studienummer 201507658 (tidligere bestod gruppen af to medlemmer, men den ene har valgt ikke at fortsætte med faget).

Krav til systemet

Mountain Goat format user stories fra projektformuleringen

- As a "Tobin Taxer" I want to tax all transaction with a charge af 1% of the total transaction value for then to be able to send the tax to the state.
- As a "Public Share OwnerControl" I want to be informed of any changes in the share ownerships for then to be able to track who is the owner of any share and series of shares.
- As a "Stock Trader Broker" I want to be the broker between the seller (provider) og the buyer (requester) for then to be able to intermediate the trade of one or more shares or one or more series of shares between one or more provideres and/or one or more requesters.
- As a "Stock Trader Broker", "Stock Share Provider" or a "Stock Share Requester", I want to inform the Tobin Tax Control, that a transaction of a certain value has been committed, for then to be able obey the public rules about taxing and paying the 1% Tobin Tax.
- As a "Stock Trader Broker", "Stock Share Provider" or a "Stock Share Requester", I want to inform the The Public Share Owner Control, that a share, many shares or a series of shares or many series of shares has changed ownership, for then to able to inform the public about who own the shares.

Use cases

Derfra kan der udarbejdes bestemte use cases, som giver bedre overblik over, hvad systemet skal kunne og hvordan systemet skal deles op i services.

Use case 1: Bruger sætter en aktie til salg

Herunder ses en fully dressed use case 1. Kun de relevante punkter er medtaget.

NAVN	BRUGER SÆTTER EN AKTIE TIL SALG
MÅL	Aktien er sat til salg
PRÆKONDITION	Bruger findes i databasen Bruger ejer aktier
HOVEDSCENARIO	1. Der vælges en bruger på forsiden 2. Inde i brugerprofilen ses oversigten over aktier 3. Der vælges hvor mange aktier, man vil sælge 4. Der trykkes på 'Set stocks'

Use case 2: Bruger køber en aktie

Herunderses en fully dressed use case 2. Kun se relevant punkter er medtaget. Der er skrevet, hvilken service er ansvarlig for bestemte hændelser. Alle services (på nær UIService) har forbindelse til hinanden gennem TransactionService

NAVN	BRUGER KØBER EN AKTIE
MÅL	Bruger køber en aktie fra en anden bruger
PRÆKONDITION	Både sælger og køber findes i databasen Sælgeren ejer aktier Køberen har råd til at købe en aktie
POSTKONDITION	Køberen har fået ejerskab over aktien Køberen har fået trukket beløbet fra kontoen Skat er blevet trukket fra beløbet Sælgeren har fået beløbet efter skat Salget er blevet noteret i Sales-databasen
HOVEDSCENARIE	<ol style="list-style-type: none">1. Der vælges en bruger på forsiden2. Inde i brugerprofilen ses oversigten over eksisterende aktier, samt mulighed for at købe flere aktier3. Der vælges at købe flere aktier4. Der ses en oversigt over tilgængelige aktier5. Der vælges antal af den aktie, man vil købe6. Der trykkes på 'Buy'7. Der tjekkes, om brugeren har råd til en aktie (TransactionService) Ext. 1: Bruger har ikke råd8. Der udregnes skat af beløbet (TaxService)9. Brugeren får trukket beløbet fra kontoen (UserService)10. Sælgeren får beløbet efter skat (UserService)11. Køber får ejerskab over aktien (StockService)12. Salget noteres (SalesService) Ext. 1: Der kommer en fejlmeddelelse "User cannot afford this stock".

Skitsen til UI

Der er blevet besluttet ikke at lave login og registrerings- funktioner til applikationen. Derfor vælger man en bruger på forsiden, som man vil agere som. Derudover er der på forsiden oversigt over seneste salg, som er blevet noteret. Forsiden ses på figur 1.

Choose a User				Latest Sales
Anders Andersen	13000 kr	Portfolio		Dennis Bülow just bought 100 'Jyske Bank' stocks from Jesper Tørresø
Leah PW	10000 kr	Portfolio		Jesper Tørresø just bought 300 'NovoNordisk' stocks from Anders Andersen
Dennis Bülow	15000 kr	Portfolio		
Jesper Tørresø	17000 kr	Portfolio		

Figur 1: Forside af applikationen

Når man vælger en bruger, kan man se, hvilke aktier de ejer, kan sætte aktier til salg, samt købe nye aktier.

Jesper Tørresø				
Company Name	Stock price	Amount of stocks	Set for sale	
NovoNordisk	218,45	500	100 ▾	<button>Sell</button>
Jyske Bank	137,18	400	200 ▾	<button>Sell</button>
<button>Buy Stocks</button>				<button>Return to main page</button>

Figur 2: Brugerprofil

Hvis en bruger trykker på 'Buy Stocks' bliver man ført til oversigten af tilgængelige aktier. Man har til enhver tid mulighed for at gå tilbage til brugerprofilen eller forsiden. Når man trykker på 'Buy' od for en aktie, bliver man ført tilbage til brugerprofilen med en opdateret aktieportfolie.

Jesper Tørresø: 17000 kr

Company Name	Stock price	Set for sale by	Available amount	
NovoNordisk	218,45	Anders Andersen	100 ▾	Buy
Jyske Bank	137,18	Dennis Bülow	200 ▾	Buy

[Return to profile](#)[Return to main page](#)

Figur 3: Køb af aktier

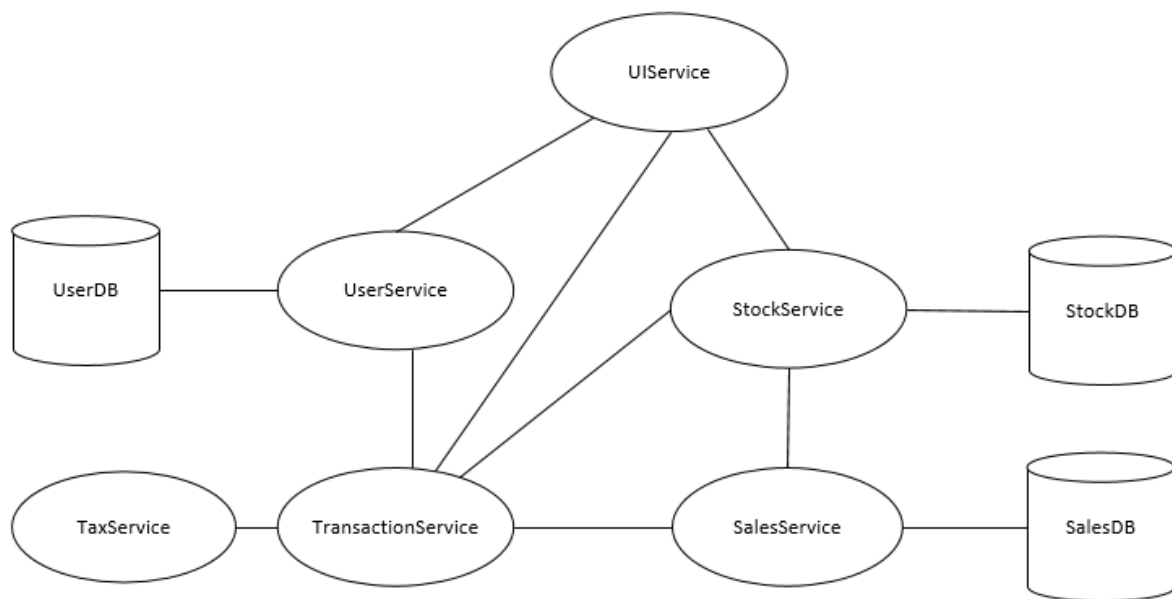
Arkitektur

Oversigt over services

På figur 4 ses en oversigt over, hvilke services systemet skal bestå af. Denne figur er lavet før implementeringen, derfor er dette bare et udkast. En endelig service-oversigt ses på figur 6.

Ansvarsområder for services er fordelt på følgende måde

- UIService er den service, som bruger har kontakt til. Denne viser oversigt over brugere i systemet, seneste salg, samt hvilke aktier en bruger har.
- StockService har ansvaret for at udføre CRUD-operationer på StockDB
- UserService har ansvaret for at udføre CRUD-operationer på UserDB
- TaxService har ansvaret for at trække skat fra en overførsel, hvilket gør den den mindste service
- TransactionService er den største service og har afhængigheder til 4 andre services. Den udfører meget af arbejdet, men har dog heller ingen storage tilknyttet



Figur 4: Oversigt over services

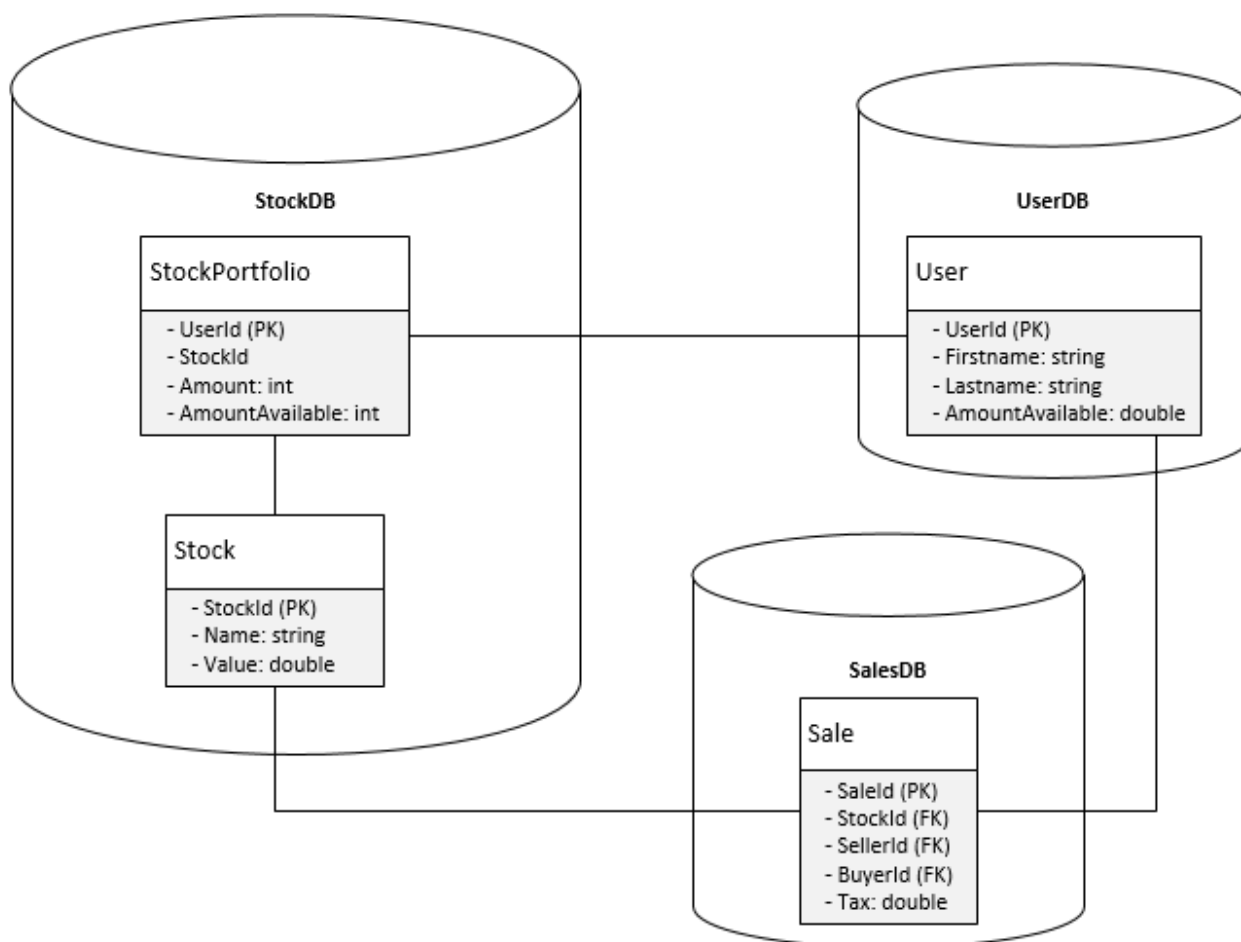
Arkitekturen er lavet efter N+1 modellen, som tager udgangspunkt i 4+1 model¹. En 4+1 model har logical, process, development og physical views (4), for hver use case (+1). N+1 er mere flydende, da den kun omfatter de views, som er relevante for produktet. N+1 omfatter derfor følgende:

- Logical view: oversigt over microservices
- Data view: systemets klasser og deres fordeling i databaser
- Implementation view: der ses, hvilke microservices systemet består af og hvordan de interagerer med hinanden. Der vises også hvordan de er deployed ift. hinanden (pods, containers mm.)
- Deployment view: der ses, hvilken metode blev brugt til at deploye microservices på GKE clusteret

Data view

På figur 5 ses, hvordan datakommunikation i systemet foregår. Der er kun 4 klasser, som dog er fordelt over 3 forskellige databaser.

¹ Se referenceliste [N+1]



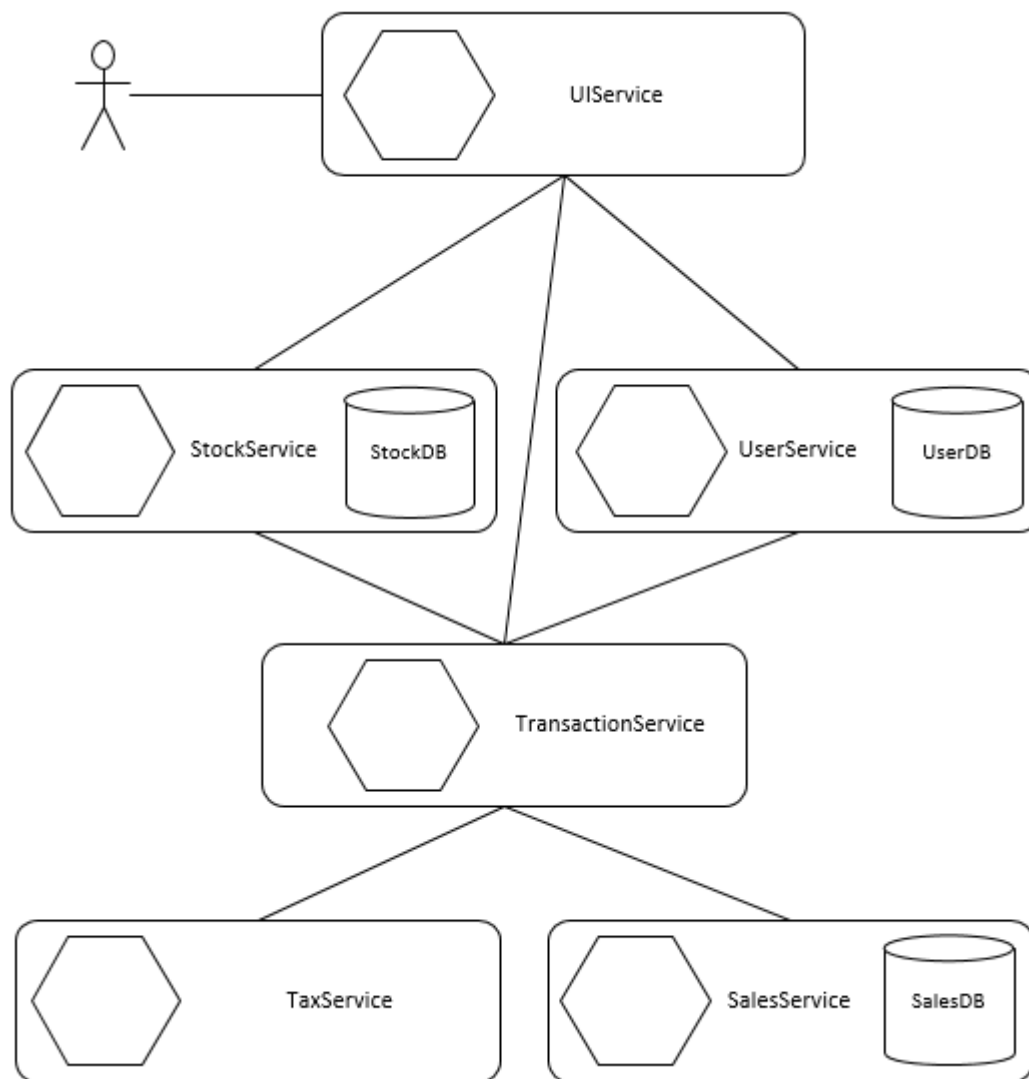
Figur 5: Datakommunikation

Implementation view

I løbet af implementeringen og deployment af systemet, viste det sig ikke at være muligt at lave en sqlserver deployment. Derfor har UserService, StockService og SalesService en *in memory database*². Det betyder, at data ikke bliver persisteret, når servicen ikke kører længere. Da services er deployed på GKE F19ITONK clusteret og ikke lokalt, er det dog ikke et problem, da disse regnes med at køre indtil eksamen er slut. Disse services har også en swagger-brugergrænseflade³, så objekterne nemt kan oprettes. Alle services på nær TaxService og TransactionService kan tilgås via external endpoints. For at se brugerfladen/teste User-, Stock- og SalesService skal der skrives /swagger efter endpointet.

² Se referenceliste [IMD]

³ Se referenceliste [SWGR]



Figur 1: Microservicearkitektur

Deployment view


Microservices er blevet deployed på GKE clusteret vha. kubectl-kommandoer via PowerShell. Figur 7 viser, hvordan TransactionService er blevet deployed. Projektet blev lavet til et image og pushed til Docker mit repository under navnet microservicetransaction. Derefter blev der brugt 2 kubectl kommandoer til at lave en deployment på GKE clusteret. Der ses, at der først skal laves en deployment, som derefter skal blive exposed. Det er vigtigt at tilføje `--namespace=*your namespace*` til sidst, så den ender det rigtige sted i stedet for default-namespace. I kubectl expose er det også vigtigt at tilføje `--type=LoadBalancer`, hvis servicen skal tilgås via external endpoints.


På figur 8 kan man se, oversigten over namespace itonk-gr19-tseis med alle microservices deployed.


```
Successfully built 13e395b0df84
Successfully tagged microservicetransaction:latest
SECURITY WARNING: You are building a Docker image from Windows against a
non-windows Docker host. All files and directories added to build context
will have '-rwxr-xr-x' permissions. It is recommended to double check an
d reset permissions for sensitive files and directories.
PS C:\Users\Leah\source\repos\TransactionService\TransactionService\dist>
docker tag microservicetransaction au547175/microservicetransaction
PS C:\Users\Leah\source\repos\TransactionService\TransactionService\dist>
docker push au547175/microservicetransaction
The push refers to repository [docker.io/au547175/microservicetransaction]
4ffc5e959a4d: Pushed
13d1cefd7f81: Layer already exists
d585b780b193: Layer already exists
ba77a4b04402: Layer already exists
29d90ae5bfcf: Layer already exists
f2cb0ecef392: Layer already exists
latest: digest: sha256:289b882d1c7a017cdd9c612104ae082d67a79377158722b1ca
a35500f246faab size: 1587
PS C:\Users\Leah\source\repos\TransactionService\TransactionService\dist>
kubectl create deployment itonk-gr19-transactionservice --image=docker.i
o/au547175/microservicetransaction:latest --namespace=itonk-gr19-tseis
deployment.apps/itonk-gr19-transactionservice created
PS C:\Users\Leah\source\repos\TransactionService\TransactionService\dist>
kubectl expose deployment itonk-gr19-transactionservice --type=LoadBalan
cer --port=8080 --target-port=80 --namespace=itonk-gr19-tseis
service/itonk-gr19-transactionservice exposed
PS C:\Users\Leah\source\repos\TransactionService\TransactionService\dist>
```

Figur 7: Deployment i PowerShell

Workload Status


Deployments


Pods


Replica Sets

Deployments

Name	Labels	Pods	Created	Images
✓ itonk-gr19-saleservice	app: itonk-gr19-saleservice	1 / 1	8 days ago	docker.io/au547175/microservicesales:latest
✓ itonk-gr19-taxingservice	app: itonk-gr19-taxingservice	1 / 1	8 days ago	docker.io/au547175/microservicetax:latest
✓ itonk-gr19-transactionservice	app: itonk-gr19-transactionservice	1 / 1	19 minutes ago	docker.io/au547175/microservicetransaction:latest
✓ itonk-gr19-uiservice	app: itonk-gr19-uiservice	1 / 1	9 minutes ago	docker.io/au547175/microserviceui:latest
✓ itonk-gr19-userservice	app: itonk-gr19-userservice	1 / 1	8 days ago	docker.io/au547175/microserviceuser:latest
✓ stockservice	app: stockservice	1 / 1	7 days ago	docker.io/au547175/microservicestock:latest

1 - 6 of 6 |< < > >|

Figur 8: Namespace itonk-gr19-tseis på GKE clusteret

Transparens i applikationen

Applikationen opfylder

1. Adgangstransparens (Grænsefladen er ens for lokale og globale services)
 2. Placeringstransparens (Bruger behøver ikke kende til den fysiske/virtuelle hosts placering, så længe brugeren tilgår applikationen via et external endpoint på Kubernetes clusteret)
 4. Replikeringstransparens (Replikation af en service bevarer synkroniseringen til sin oprindelige service)
 5. Samtidighedstransparens (Det er ikke synligt, at flere services bruger den samme service samtidigt, da alle services i backend er implementeret asynkront. Derudover er der ingen services, som deler en datastorage)
 6. Skaleringstransparens (Man kender ikke til, hvorledes omfanget af et system er opnået)
 7. Ydelsestransparens (Man kender ikke til, hvorvidt ydeevnen er opnået og opretholdt)
 8. Fejltransparens (Man ved ikke, hvordan systemet skjuler og håndterer fejl. Fejl og exceptions håndteres i deres respektive services)
- Testbarhedstransparens (Systemet kan teste sig selv for forskellige parametre)

Applikationen opfylder ikke

3. Mobilitetstransparens (Hvis services med in-memory-database bliver flyttet, vil deres data forsvinde. Dette gælder kun for SalesService, UserService og StockService). Denne regel er kun delvist opfyldt.
- Sporbarhedstransparens (Alle kald og svar til og fra systemet, både inde og ude er sporbare. Denne regel er ikke opfyldt, da loggingstransparens heller ikke er opfyldt.)
 - Loggingstransparens (System- og applikationshændelser skal kunne logges automatisk. En LogService skulle have været implementeret, som var forbundet til alle services og loggede alle hændelser. Grundet et 'mandefald' i gruppen blev denne opgave dog nedprioriteret og ikke implementeret)

Resultater

Applikationen er blevet implementeret og deployed efter krav til systemet og kan tilgås via et endpoint. På figurer 9-12 ses startside (figur 9), en brugers side, hvor han/hun kan sælge aktier (figur 10), en side, hvor brugeren kan købe aktier (figur 11), samt en side med oversigten over seneste salg (figur 12).

34.77.137.229:8080

The Stock Exchange Interaction System Home Recent sales

Choose a user

Full name	Available funds	
Leah PW	17000 kr	Choose
Dennis Bülow	18000 kr	Choose

Figur 9: Startside af applikationen

The Stock Exchange Interaction System Home Recent sales

Dennis Bülow

Company name	Stock price	Amount of stocks	Set for sale	
Jyske bank	127.17	450	<input type="text" value="200"/>	Sell stock

Buy stocks

Back to main page

Figur 10: En brugers side

The Stock Exchange Interaction System Home Recent sales

Dennis Bülow

Seller	Company name	Stock price	Amount of stocks	
Leah PW	Jyske bank	127.17	400	Buy

Back to user's profile

Figur 11: En side, hvor brugeren kan købe aktier

The Stock Exchange Interaction System Home Recent sales

Recent sales

Dennis Bülow bought 100 of Jyske bank stocks from Leah PW at Thursday, 04 June 2020 07:42:15 with 127.17 kr in tax.

Dennis Bülow bought 100 of Jyske bank stocks from Leah PW at Monday, 01 June 2020 20:09:48 with 127.17 kr in tax.

Figur 12: Oversigten over seneste salg

Forbedringer

Selvom applikationen er levedygtig og opfylder kravene i mountain goat format, har der været udfordringer med både implementeringen og deployment.

Grundet udfordringer med deployment af databaseservice, har det været nødvendigt at implementere en in-memory-database til 3 af microservices. Det er ikke et problem i sig selv, men hvis services bliver flyttet, mistes al data. En in-memory-database gør også, at reglen for mobilitetstransparens ikke opfyldes, da servicen ideelt ikke må blive flyttet.

Gruppen har også mistet et medlem, som kunne have implementeret en LogService og lavet en mere præsentabel frontend. I stedet blev disse opgaver nedprioriteret, arkitekturen måtte derfor laves om, og brugerfladen blev noget simplere. Derudover blev regler for lognings- og sporbarhedstransparens ikke opfyldt pga. manglende LogService.

Referencer

[N+1] Kruchten, P. "Architectural Blueprints—The "4+1" View Model of Software Architecture" URL:

<https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf>

[IMD] URL: <https://entityframeworkcore.com/providers-inmemory>

[SWGR] URL: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-3.1&tabs=visual-studio>