Project 4 | De Centrale Bank

# Jelle van Koppen Bytegroep 3 | NuggetsBV

Inhoudsopgave

[Jelle van Koppen 1](#_Toc515383468)

[Beveiliging: 3](#_Toc515383469)

[Salting: 3](#_Toc515383470)

[Veilig? Nee! 3](#_Toc515383471)

[Encryption: 3](#_Toc515383472)

[MD5 encryptie 3](#_Toc515383473)

[RSA Encryptie 3](#_Toc515383474)

[Veilig? Ja! 3](#_Toc515383475)

[Overzicht Centrale bank 5](#_Toc515383476)

[Ontwerp 1 5](#_Toc515383477)

[Ontwerp 2 6](#_Toc515383478)

[MQTT Protocol: 7](#_Toc515383479)

[Ontwerp 1 7](#_Toc515383480)

[Ontwerp 2 8](#_Toc515383481)

# Beheer

## Gitlab

Versiebeheer is te vinden op mijn persoonlijke GitHub.  
Deze repository bevat de code en documentatie van zowel project 3 als project 4 sinds deze projecten zo in elkaar verweven zijn.

Link: <https://github.com/JellevanKoppen/Project_3_Bank> (Of ctrl + klik op de afbeelding)

## Risicolog en issue tracking

Op GitHub zijn ook de risicolog en het issue tracking document te vinden.

Risicolog:

Issue tracking:

# Beveiliging:

Beveiliging is een van de belangrijkste kwaliteitseisen als het gaat om een bank. Hier wordt namelijk veel met persoonlijke gegevens gewerkt en worden er continu transacties gedaan met geld. Er is dus goede beveiliging nodig om deze gegevens veilig te houden. Ik heb een aantal beveiligingsmethoden onderzocht en deze methodes met elkaar vergeleken.

## Salting:

Salting is een methode waarbij berichten worden voorzien van geheimen tekens op vooraf afgesproken plekken. Alleen de ontvanger en verzender weten wat deze tekens zijn en op welke plekken deze zijn toegevoegd.   
Voorbeeld:

* Geheime tekens: €&\*
* Toevoegen aan begin, na twee letters en aan het einde

Woord: Pinautomaat

Woord met salting: €&\*pi€&\*nautomaat€&\*

### Is dit veilig?

Nee, salting van zichzelf is geen veilige methode om berichten zo te bewerken dat ze niet meer leesbaar en herkenbaar zijn voor een persoon of computer. Dit komt door de volgende redenen:

* Voor een hacker is het heel makkelijk om een woord te kunnen herkennen. De hacker kan makkelijk het patroon herkennen en ziet dus zo wat het oorspronkelijke woord zou moeten zijn geweest..
* Een computer kan filteren op tekens die steeds herhaald worden door een simpel patroonherkenningsscript te maken.
* Dit is dus niet genoeg beveiliging van zichzelf

## Encryption:

Dit breng ons bij encryptie. Encryptie wil zeggen dat een bericht wordt omgezet met behulp van een bepaald algoritme. Dit algoritme zorgt ervoor dat het bericht onherkenbaar wordt gemaakt voor zowel mens als computer.

De encrypties die ik heb onderzocht:

### MD5 encryptie

MD5 encryptie zet een bericht om in getallen en letters die voor een normaal persoon niet meer te begrijpen zijn. Er zijn geen logische woorden meer uit te halen en er is geen mogelijkheid om dit weer terug te draaien zonder dat je weet wat er uit komt. MD5 encryptie is een erg achterhaalde vorm van encryptie. Veel encrypties zijn al gekraakt. Een website beweerd dat het al 829.726 miljard MD5 hashes heeft gekraakt. Met behulp van salting zal de kans dat de hash al bestaat drastisch afnemen, maar er is een ander probleem.  
Deze optie is niet handig om te gebruiken voor dit project. Het is namelijk zo dat de server het bericht moeten kunnen ontcijferen omdat het bericht een instructie bevat voor de server die die moet uitvoeren. MD5 encryptie is dus niet de juiste encryptie voor dit project

Wat we nodig hebben is twee richting encryptie: Een encryptie die een bericht kan beveiligen, maar ook ditzelfde bericht weer kan ontcijferen met een specifieke vooraf afgesproken sleutel.

## RSA Encryptie

RSA Encyptie is een encryptie dat werkt met sleutels. Eén sleutel om het bericht te versleutelen (public key) en één sleutel om hetzelde bericht weer te ontsleutelen (private key). De public key zal aan elke bank worden uitgedeeld en de private key zal nooit de server verlaten. Dit komt omdat deze sleutel niet mag worden gedeeld omdat dan iedereen het bericht zal kunnen ontsleutelen die de private key in handen krijgt. Deze sleutels kunnen gegenereerd worden met bepaalde RSA key generators. Een voorbeeld hiervan vind je met de link onderaan deze pagina. De versleuteling zal 2048-bits zijn zodat het heel erg lang duurt voordat deze private key geraden is. Op de volgende pagina vind je een voorbeeld van de sleutels

Daarnaast zal de centrale server een andere private key krijgen dan de banken. De banken krijgen hun eigen private key om de berichten te ontcijferen die de server stuurt die zijn versleutelt met een andere public key. Dit is zodat wanneer een van de banken wordt gehacked. Niet het hele systeem gehacked kan worden. De private key van de centrale server zal nooit ergens anders staan dan op de server.

<http://travistidwell.com/jsencrypt/demo/>

Voorbeeld van een RSA Private Key:

-----BEGIN RSA PRIVATE KEY-----



-----END RSA PRIVATE KEY-----

Voorbeeld van een RSA Public Key:

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCYKaqdTtdkwlWcDldlftO2Adn3wEjWJwNcINln495tDTlKITKbCr6eX/mgjN/92OjKR90Wn+PQbK+ZuFlZeDjPuergqqSvYKvEXre+bgfW6IK7a+2s3uFNez2T6YcgK95EjOlUc6KLKJ9MFeCPucuS6AqLE2padNvZxJrB/O1rTwIDAQAB

-----END PUBLIC KEY-----

Zoals je ziet is er door een normaal persoon geen touw meer aan vast te knopen hoe dit precies werkt. Een computer zal de private key kunnen gebruiken om public versleutelde berichten te kunnen ontsleutelen.

# Privacy

Net als beveiliging is privacy ook een van de grootste kwaliteitseisen van een bank. Aangezien een bank gigantisch veel gegevens over je verzameld om bijvoorbeeld identiteitsfraude te voorkomen, moet er goed worden nagedacht over hoe deze informatie verwerkt moet worden en in hoeverre deze informatie opgeslagen mag worden. Het is niet de bedoeling dat derden informatie krijgen tot de bankgegevens van de klanten. Dit gaat in tegen de privacywetgeving.

Er moeten duidelijke regels worden opgesteld over welke informatie mag worden gedeeld en welke informatie goed en veilig moet worden opgeslagen in de database zodat niemand hierbij mag.

Hieronder heb ik een aantal persoonsgegevens onderzocht:

## Persoonlijke gegevens

Persoonlijke gegevens zijn verboden om te delen met derden zonder de toestemming van de gebruiker. De bank zal persoonlijke gegevens moeten bewaren om onder anderen identiteitsfraude te voorkomen en om ervoor te zorgen dat wanneer de bank geld uitleent, deze persoon ook daadwerkelijk bestaat en de lening kan terugbetalen. Hiervoor zal een check nodig zijn. Een check om te controleren of deze persoon wel in staat is om een lening te nemen. Deze checks kan de bank nooit allemaal zelf aan. Hiervoor zal dus bijvoorbeeld een derden nodig zijn om deze checks te doen.   
Om dit te kunnen doen zal bij het aanmaken van een account een privacy statement moeten worden geaccepteerd waarbij de klant accepteert dat de bank deze gegevens mag gebruiken om de achtergrondcheck te doen.   
Wanneer de bank deze gegevens niet bewaart, dan zal dit ten kosten gaan van de veiligheid en betrouwbaarheid van de bank. De bank zal deze gegevens echter wel goed moeten bewaren.  
Wanneer de klant weg gaat bij de bank en geen lening meer heeft en dus in geen manier meer verbonden is met de bank. Dan zal de bank ook de gegevens van de ex-gebruiker moeten verwijderen. De bank heeft niks meer aan deze gegevens en als deze gegevens in de database blijven staan deze gegevens onnodig bloot aan veiligheidsbreuken. Aangezien banken altijd een gewild doelwit zijn voor hackers, mogen deze gegevens niet onnodig rond blijven slingeren in de database.

## Bank transacties

Uit elk bank transactie valt wel een verhaal te halen. Zo kun je, wanneer je alle transacties combineerd, makkelijk een patroon herkennen van hoe de gebruiker zijn manier van leven is. Deze informatie is erg gewild bij derde partijen. Deze partijen kunnen voorspellen wat, wanneer en waar je iets koopt. Zo kunnen ze bijvoorbeeld op het onderbewustzijn inspelen en met slimme marketingtrucks de gebruiker overhalen om iets te ondernemen. Deze gegevens zijn daarom erg gevoelig en moeten worden besproken als het gaat om privacy.  
Bank transacties worden met velen tegelijk gedaan en moeten dus snel kunnen worden weggeschreven en mogen niet teveel tijd kosten. Bank transacties zijn belangrijk om bewaard te worden wanneer bijvoorbeeld: Iemand verdachten aankopen maakt, zodat de eigenaar kan worden gewaarschuwd. Of wanneer iemand zijn afschriften wil zien, dan moeten deze transacties aan de gebruiker worden getoond.  
Maar de enige die deze transacties mogen zien, zijn gebruiker zelf en de beveiligingssystemen van de bank. Elke andere vorm van het delen van deze transacties zonder toestemming van de gebruiker zal worden gezien als een inbraak op de privacy. Hiermee moet dus goed rekening mee gehouden worden, dat deze gegevens niet op straat komen te liggen.

# Effectiviteit

Effectiviteit gaat allemaal om dat het eindproduct moet doen wat het moet doen. In dit onderdeel zal ik vertellen hoe ik verschillende componenten van het banksysteem heb onderzocht en ben ik op zoek gegaan naar de meest effectieve manier om dit project te kunnen laten slagen.

## Gegevens uitwisseling

Om de banken met elkaar te kunnen laten communiceren is gegevens uitwisseling nodig. Banken zullen informatie met elkaar moeten delen om het hele process te kunnen laten werken.  
Er zijn meerdere manieren om dit aan te pakken.

## MQTT:

MQTT staat voor Message Queuing Telemetry Transport. Het is een protocol waarmee computers (of in dit geval banken) zich kunnen abonneren op kanalen. Ze kunnen ook publiceren op kanalen. Dat wil zeggen dat ze berichten kunnen plaatsen op een kanaal. En iedereen die met dit kanaal is verbonden kan dan dit bericht lezen. Uiteraard moet iedereen ook verbonden zijn met dezelfde server om dit te laten werken. Dit is een handige maar onveilige methode. Handig omdat dit heel makkelijk toestaat om berichten van de ene bank naar de centrale bank en weer terug te sturen door middel van kanalen. Alleen wel onveilig omdat al deze berichten gewoon als ‘plain text’ worden verstuurd. Implementatie van dit protocol zal enige extra beveiliging eisen.

### DDS:

DDS staat voor Data Distribution Service. Het gebruikt dezelfde publish-subscribe methode als MQTT. Het DDS protocol wordt vooral gebruikt in grote bedrijven waarbij duizenden berichten tegelijk sturen. DDS heeft ook een QoS (Quality of Service) optie. Deze optie zorgt ervoor dat berichten met de hoogste prioriteit als eerste en altijd aankomen. Het verschil met DDS en MQTT is dat met DDS er meerdere kanalen in kanalen aangemaakt kunnen worden. Het is een stuk ingewikkelder om DDS aan de praat te krijgen dan MQTT. DDS is vooral geschikt voor de grotere projecten.

# Vertrouwen

## Betrouwbaarheid

Een systeem is betrouwbaar als het eindproduct een lange tijd blijven voldoen aan de functionele en niet-functionele systeemeisen. Betrouwbaarheid is dus te meten in hoe vaak er onderhoud zal moeten worden gepleegd aan het systeem.

## Updaten

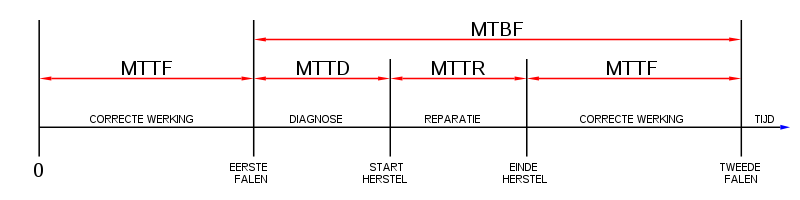
De protocollen die worden gebruikt, de software van de databases of de servers, zullen regelmatig moeten worden geupdate. Wanneer dit niet gebeurt zorgt dit voor beveiligingsouten en verliest het systeem hiermee ook zijn betrouwbaarheid. Er zal zeker maandelijks onderhoud moeten worden gepleegd met het installeren van de laatste updates van het hele systeem. Als het systeem hierin slaagt en geen andere falen geeft, dan zal het vertrouwen in het systeem hetzelfde blijven.

## MTBF

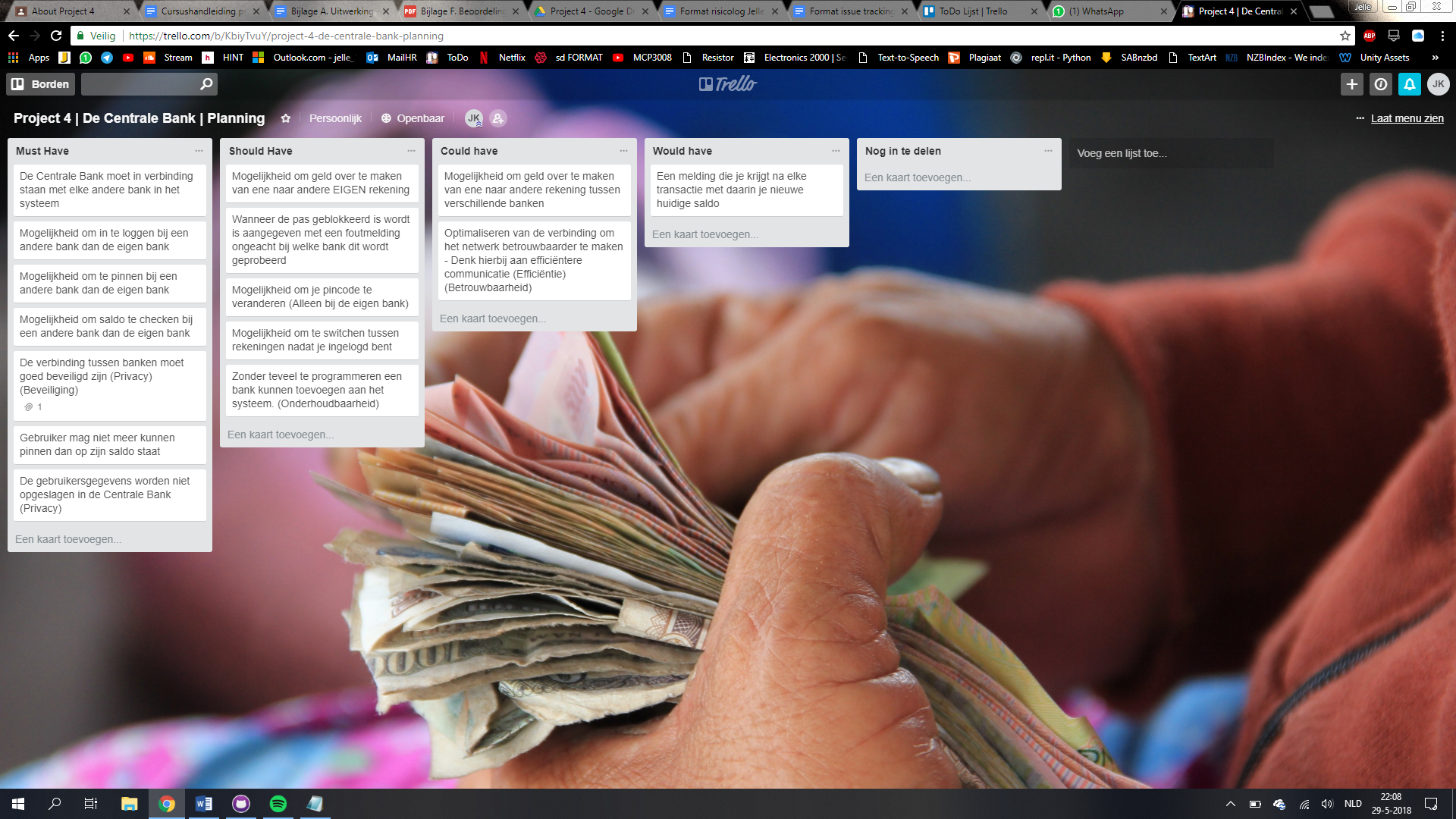
MTBF betekend Mean time between failures. Dit staat voor de gemiddelde tijd tussen falen van het systeem. Hiermee worden vooral storingen bedoeld. Het is een term die veel in de bedrijfskunde gebruikt wordt en het wordt berekend door de gemiddelde tijd te nemen tussen falen. Dus de tijd tussen wanneer een systeem gefaald is, gerepareerd is, weer correct werkt en weer faalt. Dit is een goede indicator voor het berekenen van een betrouwbaar systeem. Het aantal uur wat een systeem kan draaien zonder falen zegt al veel over de betrouwbaarheid van het systeem.

# MTTD, MTTR, MTTF

Deze afkortingen staan respectievelijk voor: gemiddelde diagnosetijd, gemiddelde repareertijd en gemiddelde correcte werktijd. Wanneer een systeem goed ontwikkeld is zal de diagnose niet zo lang moeten duren omdat al snel duidelijk is waar het systeem gefaald is. De gemiddelde repareertijd hangt volledig af van wat en hoe het systeem heeft kunnen falen. En de correcte werktijd geeft een goede indicatie hoelang het systeem draaiend is geweest ten opzichte van dat het niet draaide.



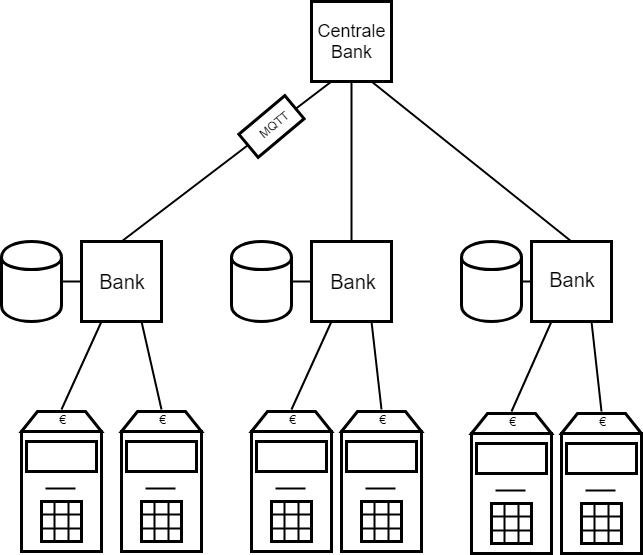
# Prioriteren van functionele eisen

De fucntionele eisen zijn geprioriteerd in een trello.  
De actuele trello planning is hier te vinden:   
<https://trello.com/b/KbiyTvuY/project-4-de-centrale-bank-planning>  
Hieronder staat de trello-planning van het moment van schrijven

# Ontwerpen

## Ontwerp 1

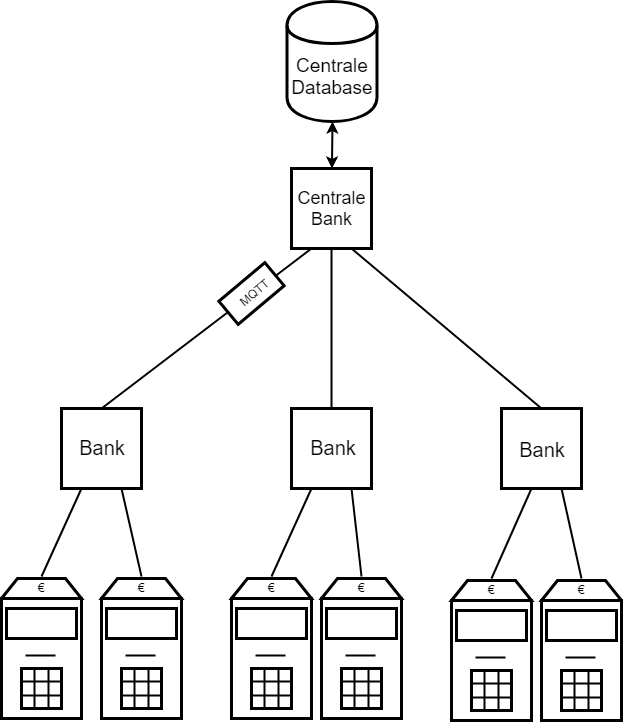
In dit ontwerp is de centrale bank een doorgeefluik. De centrale bank ontvangt aanvragen voor gegevens en stuurt deze door naar elke bank als een soort broadcaster. Er wordt dus geen gevoelige informatie opgeslagen in de centrale bank. Wanneer een bank informatie heeft dan stuurt deze dit weer terug naar de centrale bank. Deze gegevens stuurt de centrale bank weer naar de bank die op deze gegevens zit te wachten. Dit zal gebeuren volgens het MQTT protocol, waar verder in dit rapport ook uitgebreid verslag over wordt gedaan.  
Elke bank heeft zijn eigen database. De gegevens van andere accounts zullen dus moeten worden opgehaald uit de andere bank databases, via de centrale bank. Wanneer een van de banken een storing heeft zal dit geen groot probleem zijn voor het netwerk, omdat elke bank draait op zijn eigen database. Dus het gevolg zal zijn dat iedereen die bij die kapotte bank zijn account heeft, niet zal kunnen inloggen voordat de bank weer gerepareerd is. De rest van de banken zal gewoon blijven functioneren naar behoren.



Dataflow diagram

Ontwerp 2

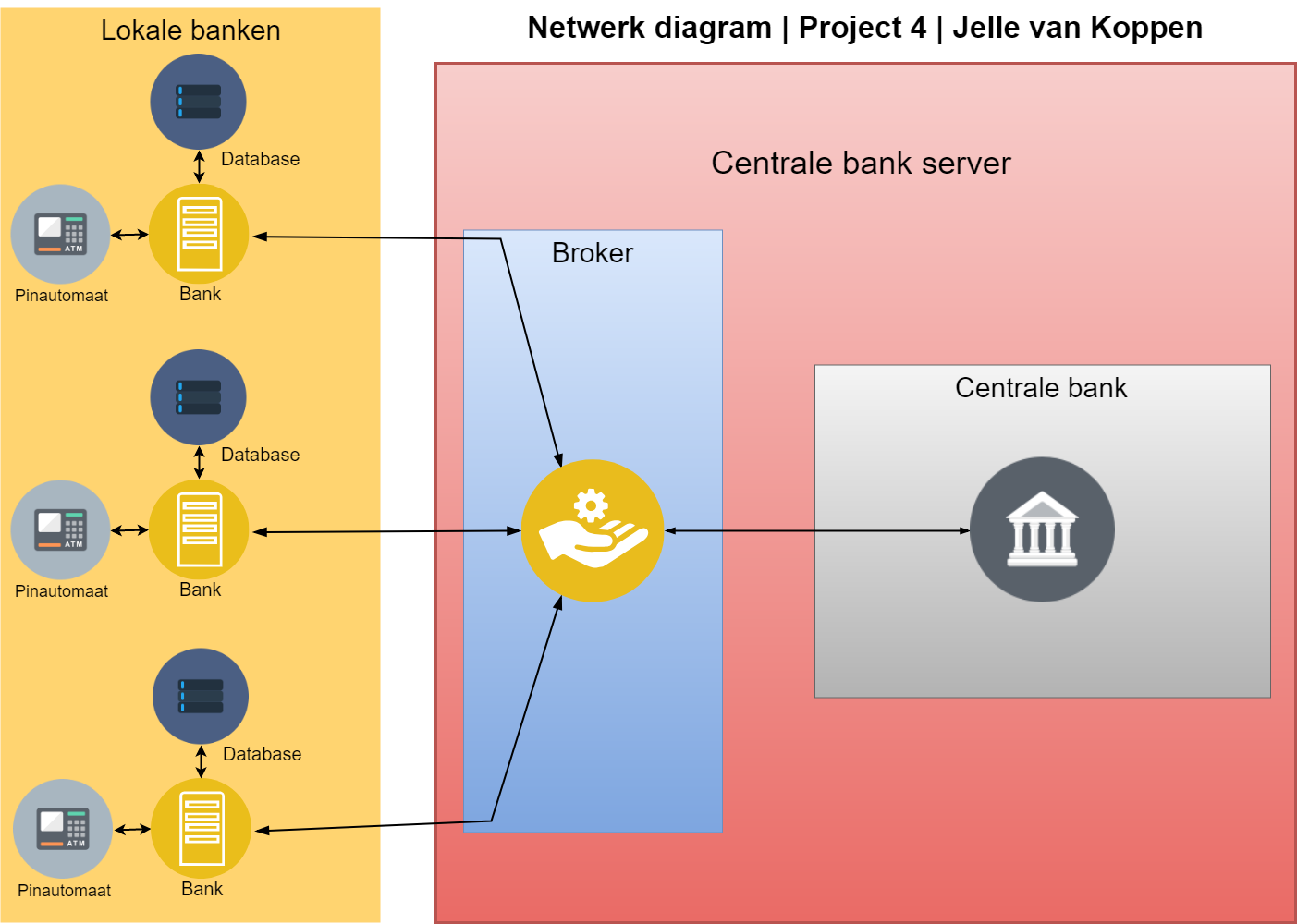
In dit ontwerp is de centrale database het kloppende hart van het hele netwerk. Hierbij bewaart de centrale database alle gegevens. De centrale bank zorgt voor de koppeling tussen alle banken. De banken bevatten geen informatie en vragen alle gegevens op bij de centrale database. Het voordeel hiervan is dat banken altijd weten waar ze de informatie kunnen vinden en dat dit volledig afgehandeld kan worden door de centrale bank zonder dat er andere banken aan te pas komen. Het nadeel is wel dat wanneer de centrale database het niet meer doet of een storing heeft, gelijk het hele systeem wordt ondermijnt. Er zullen geen transacties meer mogelijk zijn omdat er geen verbinding meer met de database is, die alle gegevens van alle accounts heeft. Dit ontwerp maakt ook gebruik van het MQTT protocol net zoals het andere ontwerp.



Dataflow diagram

|  |  |  |
| --- | --- | --- |
| **Deelpunten** | **Ontwerp1** | **Ontwerp2** |
| Betrouwbaar | X |  |
| Goedkoopst |  | X |
| Efficiënts | X |  |
| Overdraagbaar | X |  |
| Snelheid |  | X |

## Ontwerp 1 | Netwerkdiagram



# MQTT Protocol:

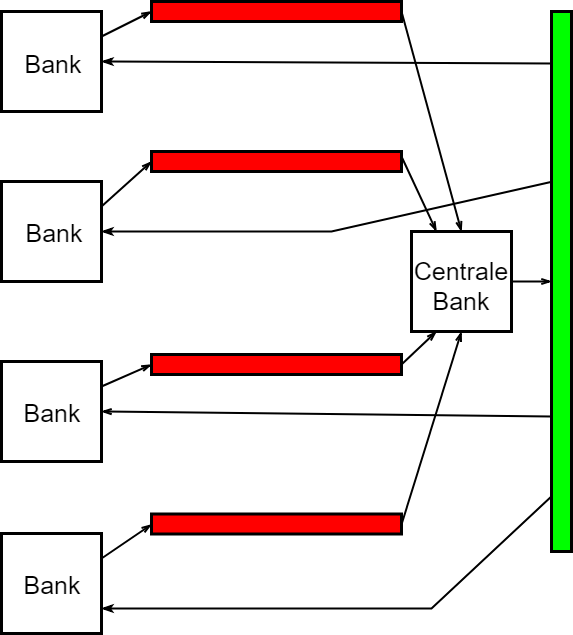
## Ontwerp 1

Elke bank heeft zijn eigen kanaal. De bank praat alleen maar in zijn eigen kanaal en luistert hier niet naar. De bank luistert alleen naar het kanaal van de Centrale bank.

De Centrale bank luistert naar elk kanaal. De Centrale bank praat alleen in zijn eigen kanaal en elke bank luistert naar dit kanaal. De centrale bank hoeft dus niet meer aangepast te worden als er een bank wordt toegevoegd aan het systeem. Dit komt doordat MQTT een zogenaamde wildcard heeft wat eruit ziet als dit: ‘#’ symbool. Hiermee kan de centrale bank zich op elk kanaal abonneren binnen het netwerk en heeft het geen update nodig wanneer een nieuw bank wordt toegevoegd.

Kanaal van Centrale bank is groen weergegeven

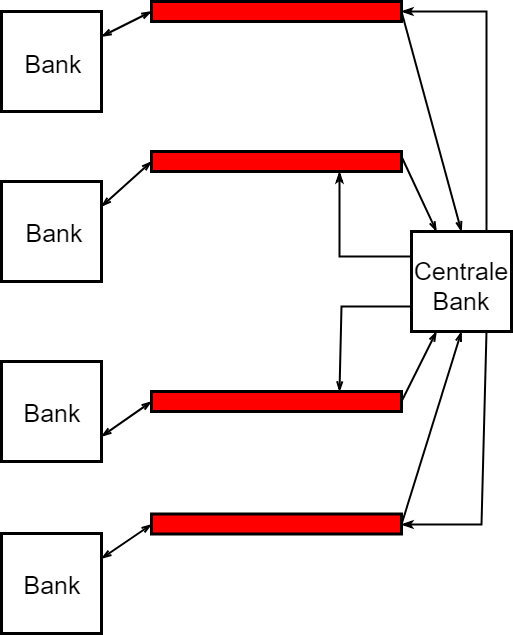
Kanalen van de banken is weergegeven in het rood



## Ontwerp 2

Elke bank heeft zijn eigen kanaal, maar in dit ontwerp heeft de centrale bank geen eigen kanaal. De centrale bank praat nu naar 1 bank tegelijk in het kanaal van die bank. De bank praat in zijn eigen kanaal en luistert hier ook naar. Dit wil zeggen dat elk kanaal nu tweerichtingsverkeer is. Dit vergt een andere aanpak van programmeren want nu zal voor elke toevoeging van een bank, de centrale bank zijn code moeten aanpassen. Wanneer de centrale bank gegevens op wil vragen van alle andere banken, dan zal het elk kanaal stuk voor stuk af moeten gaan om de gegevens te verzamelen.

Kanalen van de banken zijn weergegeven in het rood



|  |  |  |
| --- | --- | --- |
| **Deelpunten** | **Ontwerp1** | **Ontwerp2** |
| Overzichtelijk | X |  |
| Hoeveelheid kanalen |  | X |
| Verkeer per kanaal | X |  |
| Makkelijk te maken/programmeren | X |  |
| Dynamisch | X |  |
| Kanalen hebben maar één ontvanger |  | X |
| Overdraagbaarheid | X |  |

# Adviseren

Mijn advies gaat uit naar ontwerp 1. Dit geld voor zowel de centrale bank als voor de toepassing op het MQTT protocol.

## Centrale bank

In het ontwerp van de centrale bank is efficiëntie erg belangrijk. Als je kijkt naar ontwerp 2 dan zie je dat er maar één database draait. Deze database zou dan héél veel verkeer aan moeten kunnen en zou een minimale MBTF (gemiddelde tijd tussen falen) moeten hebben. Wanneer het systeem eruit ligt kan niemand in het systeem bankieren.   
In ontwerp 1 is het duidelijk dat elke bank zijn eigen lokale database heeft draaien. Dit zorgt voor meer efficiëntie omdat ten eerste, wanneer een gebruiker bij zijn eigen bank pint, er helemaal geen gegevens naar de centrale database gestuurd hoeven worden. Dit is efficiënter dan wanneer eerst contact met de centrale bank gelegd moet worden. Daarnaast is het zo dat het hele systeem betrouwbaarder is. Wanneer er een bank een storing heeft, zullen alleen de gebruikers van die bank niet kunnen pinnen. Voor de rest draait het systeem gewoon door op de banken die over zijn. Wanneer elke bank zijn eigen database heeft en er wordt een bank toegevoegd aan het systeem, dan is dit met ontwerp 1 het meest efficiënt. Dit komt doordat nu alleen de verbinding tussen bank en centrale bank gelegd hoeft te worden en de bank moet zich houden aan de voorgeschreven protocollen. De centrale bank hoeft hier dus niks voor te doen. Wanneer je met ontwerp 2 een bank wilt toevoegen zorgt dit voor meer problemen. In de database zal namelijk een extra tabel moeten worden aangemaakt en alle gegevens die de bank heeft zal moeten worden overgezet naar de database van de centrale bank. Daarom is ontwerp 1 de beste keuze.

## MQTT

Voor MQTT is het vooral belangrijk dat elke bank dezelfde regels met elkaar afspreekt. Maar willen we informatie gescheiden houden en verkeer maar één kant op laten gaan om zo min mogelijk verwarring te krijgen in de berichten, dan is ontwerp 1 de beste keuze. Met de toevoeging van één kanaal is het hele systeem al een stuk overzichtelijker. Hiermee wordt namelijk al het verkeer wat de centrale bank terug stuurt in een apart kanaal gezet. Het voordeel hiervan is dat de centrale bank met één bericht alle banken kan bereiken zonder eerst elke bank af te gaan en een persoonlijk bericht te sturen.

## Beveiliging

Voor de beveiliging is het belangrijkste dat geen hacker of computer het bericht kan ontcijferen. Hiervoor adviseer ik de RSA encryptie. Het grootste voordeel van RSA is dat het weer de ontcijferen is met de private key. Deze sleutel wordt goed opgeborgen op de server en om de berichten nog extra te encrypten zal er salt aan worden toegevoegd voor de RSA encryptie die ook geheim zullen blijven voor de rest van de wereld.

## Privacy

Wat betreft privacy adviseer ik dat de maatregelen genoemd in de analyse ook daadwerkelijk worden uitgevoerd. De banken moeten de persoonlijke informatie goed bewaren en mogen deze alleen bewaren zolang de gebruiker een klant is bij de bank. Daarnaast is het ook niet toegestaan om informatie te delen met derden zonder toestemming van de klant.