**ARCHITECTUUR DOCUMENT**

Internetbankieren

Bram Segers

Gijs Hendrikx

# Inhoud

[1. Eisen 3](#_Toc422650916)

[Functionele eisen 3](#_Toc422650917)

[Niet functionele eisen 3](#_Toc422650918)

[2. Component diagram 4](#_Toc422650919)

[3. Deployment diagram 5](#_Toc422650920)

[4. Klassediagram 6](#_Toc422650921)

[Klassediagram v1 6](#_Toc422650922)

[Sequence diagrammen v1 7](#_Toc422650923)

[Klassediagram v2 10](#_Toc422650924)

[Sequence diagram v2 11](#_Toc422650925)

[Klassediagram v3 12](#_Toc422650926)

[Klassediagram v4 (definitief) 14](#_Toc422650927)

[5. Realisatie niet functionele eisen 16](#_Toc422650928)

[6. Testen 17](#_Toc422650929)

[7. Niet gerealiseerde doelstellingen 18](#_Toc422650930)

# Eisen

## Functionele eisen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Naam | Omschrijving | Belang | Urgentie | Use Case ID |
| F.1 | Registreren | Een klant kan zich registreren | Must | Hoog | UC1 |
| F.2 | Sessie starten | Een klant kan een sessie starten | Must | Hoog | UC2 |
| F.3 | Rekening Openen | Een klant kan een rekening openen | Must | Hoog | UC3 |
| F.4 | Saldo inzien | Een klant kan het saldo van een van zijn rekeningen inzien | Must | Hoog | UC4 |
| F.5 | Geld overmaken | Een klant kan geld overmaken van een van zijn rekeningen naar een andere rekening | Must | Hoog | UC5 |
| F.6 | Transacties inzien | Een klant kan de transacties van een van zijn rekeningen inzien | Should | Laag | UC4 |
| F.7 | Sessie beëindigen | Een klant kan zijn sessie beëindigen | Should | Midden | UC6 |

## Niet functionele eisen

Voor het systeem gelden de volgende kwaliteitseisen:

Beschikbaarheid

* Als een nieuwe bank zich registreert bij de centrale bank, zijn alle rekeningen van deze bank direct beschikbaar voor overboekingen.
* De centrale bank is maximaal 48 uur per jaar en 12 uur per etmaal niet beschikbaar.
* Elke bank is maximaal 168 uur per jaar en 16 aansluitend per dag niet beschikbaar.

Performance

* Overboekingen worden binnen 5 minuten verwerkt en zichtbaar gemaakt voor beide rekeninghouders.
* Elke bank kan service verlenen aan minimaal 1000 clienten tegelijk.

Persistentie

* Alle banken gebruiken een MySQL database. Banken zijn zelf verantwoordelijk voor backups van de database.

Veiligheid

* Al het webverkeer verloopt via het https protocol.
* Een sessie wordt na 5 minuten inactiviteit automatisch uitgelogd.
* Wachtwoorden worden versleuteld opgeslagen.

Betrouwbaaheid

* Uit de unit tests moet af te leiden zijn dat 100% van de code getest is.

Usability

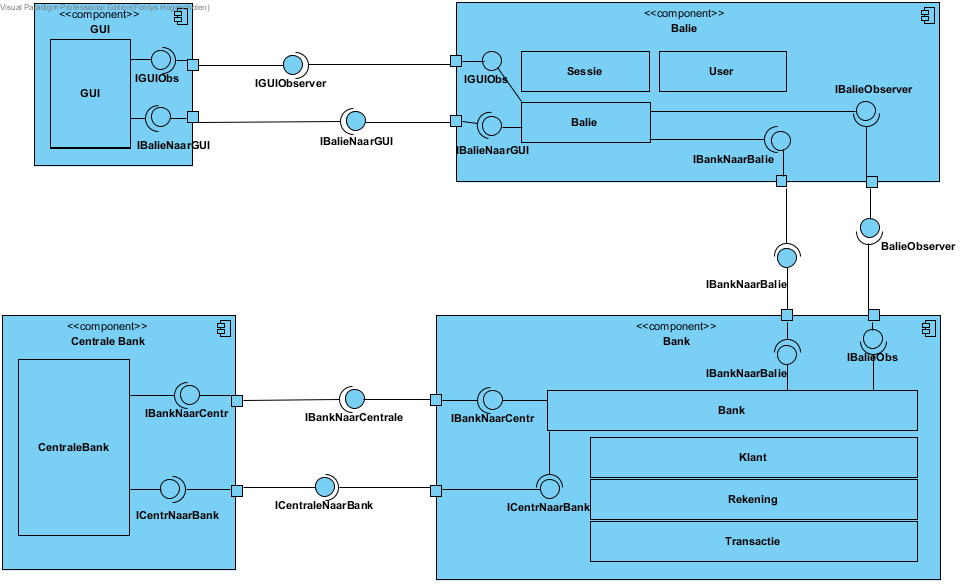
* De interface voor de rekeninghouders is gebruiksvriendelijk en intuïtief.

# Component diagram

Voor deze module gaan we uit van 4 modulaire systemen:

* De GUI. Deze verzorgt de interactie met de gebruiker. Deze interactie wordt gerealiseerd via een webpagina in een browser. De GUI biedt IGUIObserver als interface aan de balie aan.
* De balie. Deze verzorgt de interactie tussen de bank en de gebruiker. Interactie wordt gerealiseerd via een webserver. De balie verzorgt de interface IBalieNaarGUI voor de GUI en IBalieObserver voor de bank.
* De Bank. Iedere bank heeft zijn zijn eigen applicatie draaien waarin transacties worden geregistreerd en waar persistentie gewaarborgd is. De bank verzorgt de interfaces IBankNaarBalie voor de balie en IBankNaarCentrale voor de centrale bank.
* De Centrale Bank. Hiervan is één instantie. Alle banken registreren zich bij deze instantie, zodat de centrale bank alle banken kent. De centrale bank verzorgt de interface ICentraleNaarBank voor de bank.

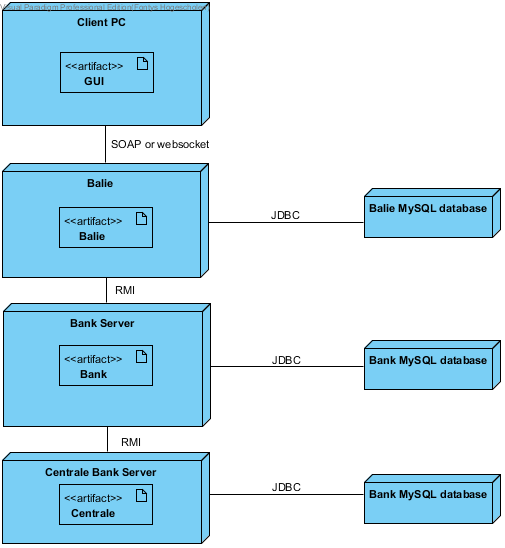
Het component diagram ziet er dan als volgt uit. Te zien is welke klassen de genoemde interfaces aanbieden of gebruiken.



# Deployment diagram

Onderstaand deployment diagram toont de (mogelijke) fysieke apparaten waarop het systeem zal draaien en de manier van communicatie tussen deze apparaten.

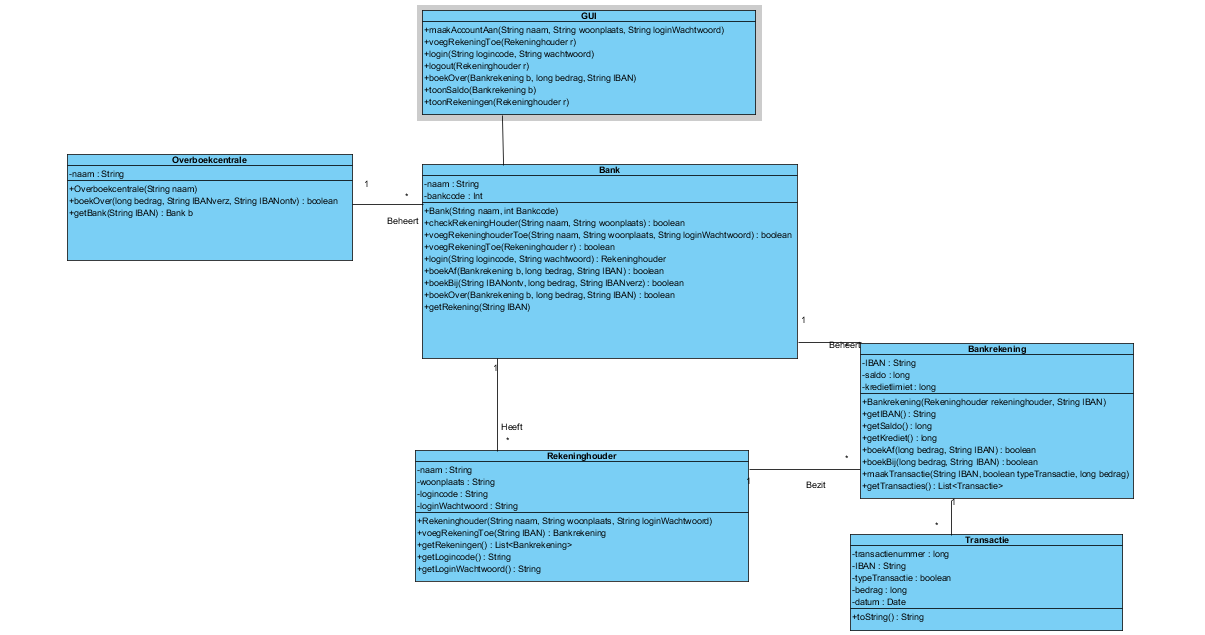
* Tussen de client PC en de Balie server zal via SOAP of websockets gecommuniceerd worden. Nader onderzoek zal duidelijk maken welke communicatiemethode de voorkeur heeft.
* De balie server communiceert met de bank server middels RMI. Tevens leest en schrijft de Balie gegevens van en naar een MySQL database middels een JDBC driver.
* De bank server communiceert verder nog met de centrale bank server via RMI en met een MySQL database middels een JDBC driver.
* De Centrale Bank communiceert met een MySQL database middels een JDBC driver.



# Klassediagram

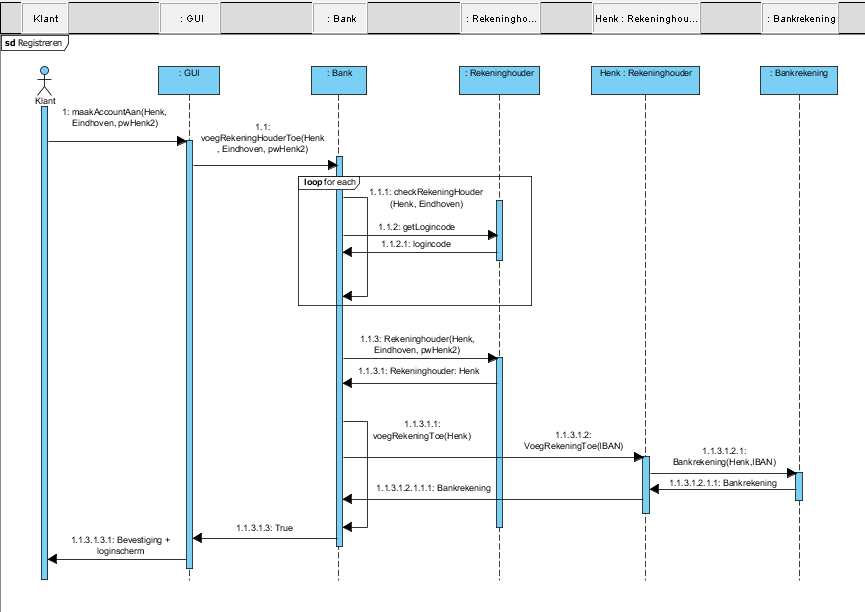
Onderstaand wordt de ontwikkeling van het klassediagram met meerdere versies aangetoond. De belangrijkste functies worden met een sequence diagram getest.

## Klassediagram v1



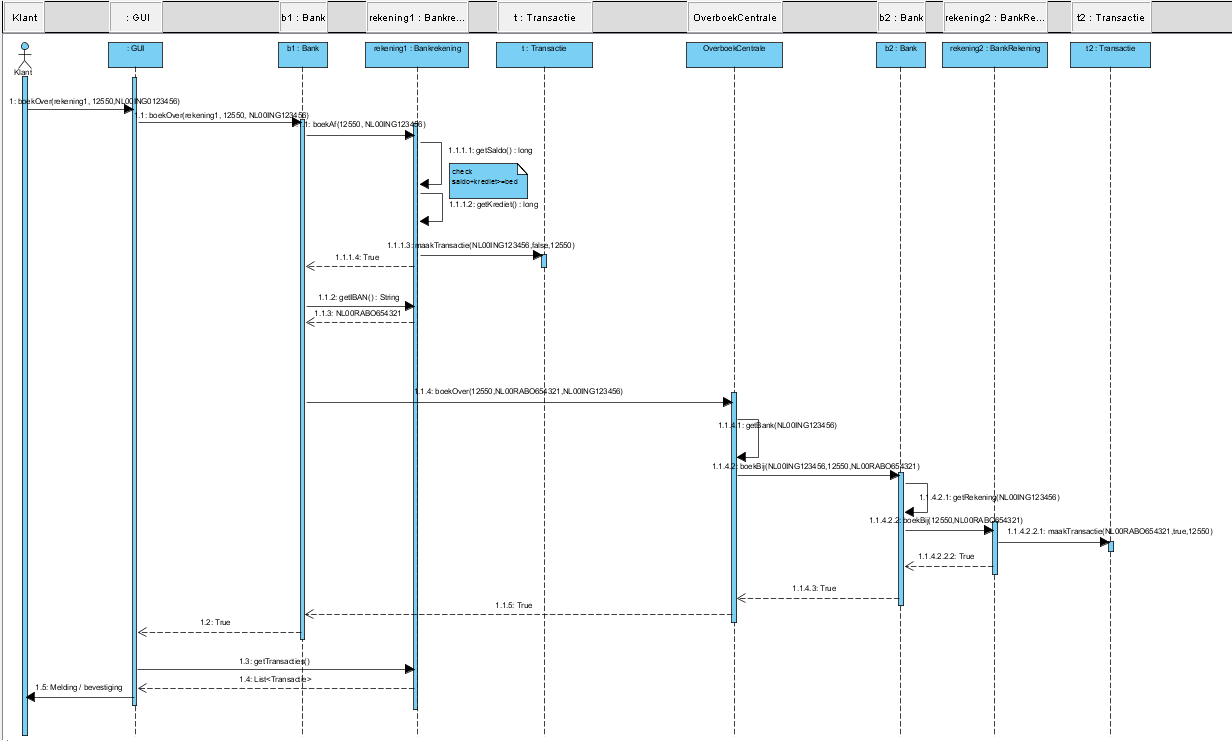
## Sequence diagrammen v1

**F.1 Registreren**



Van Bank naar GUI moet nog de (eenmalig verstrekte??) logincode worden teruggegeven, deze komt dan mee in uiteindelijke melding / bevestiging.

**F.5 Geld overmaken**

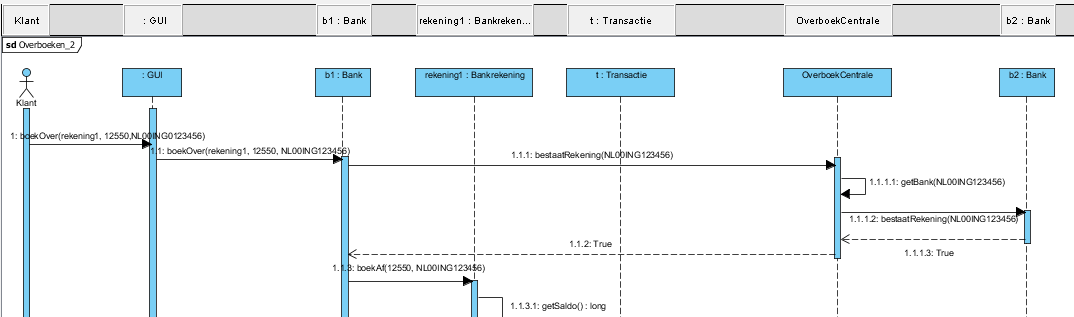


Tijdens het maken van deze sequence diagrammen kwamen we tot de conclusie dat deze eerste opzet zal moeten worden aangepast. Zoals hierboven beschreven, kan er bij Klant een bedrag worden afgeboekt en een bijbehorende Transactie worden aangemaakt. Het bijboeken op de tegenrekening kan daarna echter mislukken, omdat:

1. het opgegeven IBAN nummer niet aan een bank gekoppeld kan worden (1.1.4.1)

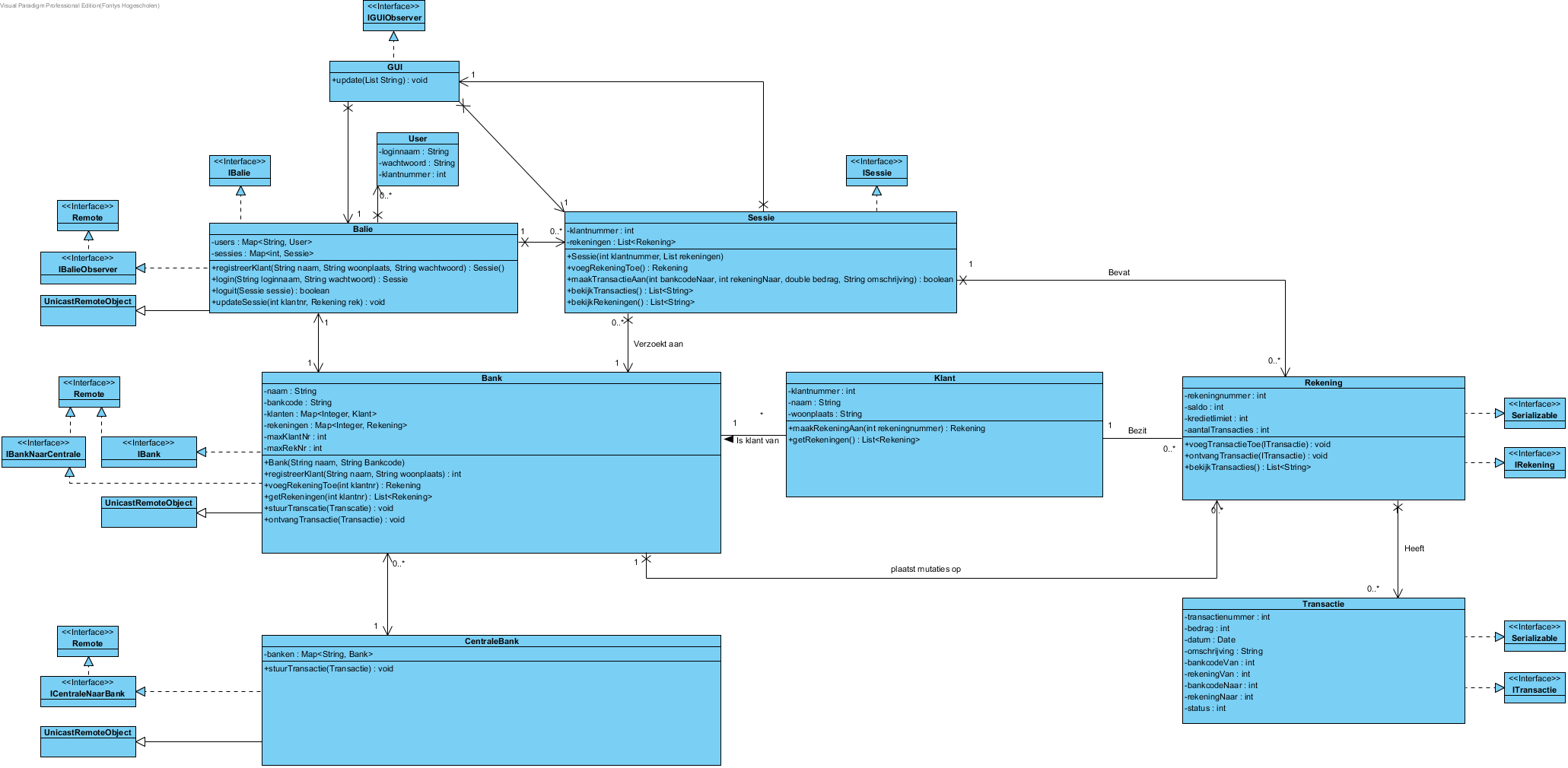
2. het opgegeven IBAN nummer niet bij de bijbehorende bank bekend is (1.1.4.2.1)

Om dit probleem op te lossen, hebben we het klassendiagram aangepast. Methode bestaatRekening() is aan Overboekcentrale en aan Bank toegevoegd. Het bovenste deel van het sequence diagram ziet er dan zo uit:



## Klassediagram v2

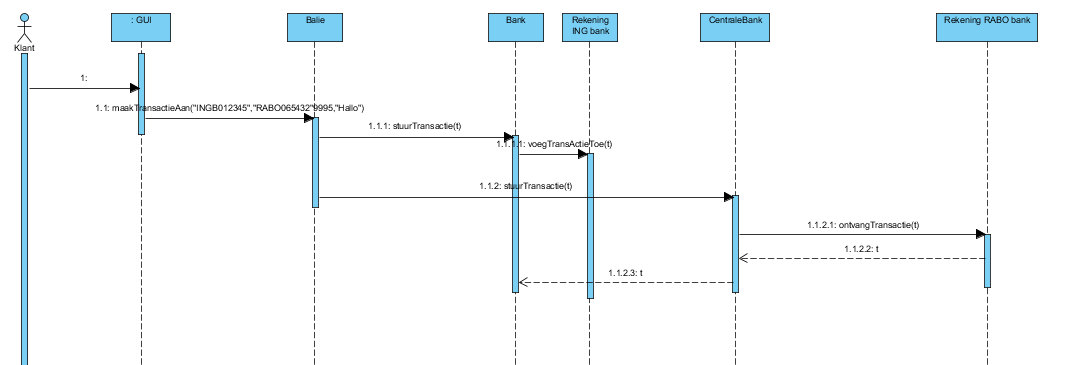
Onderstaand klassediagram toont het actuele ontwerp van het internetbankieren systeem. Om het diagram overzichtelijk te houden zijn de interfaces (en de koppelingen ernaar) beperkt gehouden. De interfaces worden verder in dit document in Javadoc verder gespecificeerd.



## Sequence diagram v2

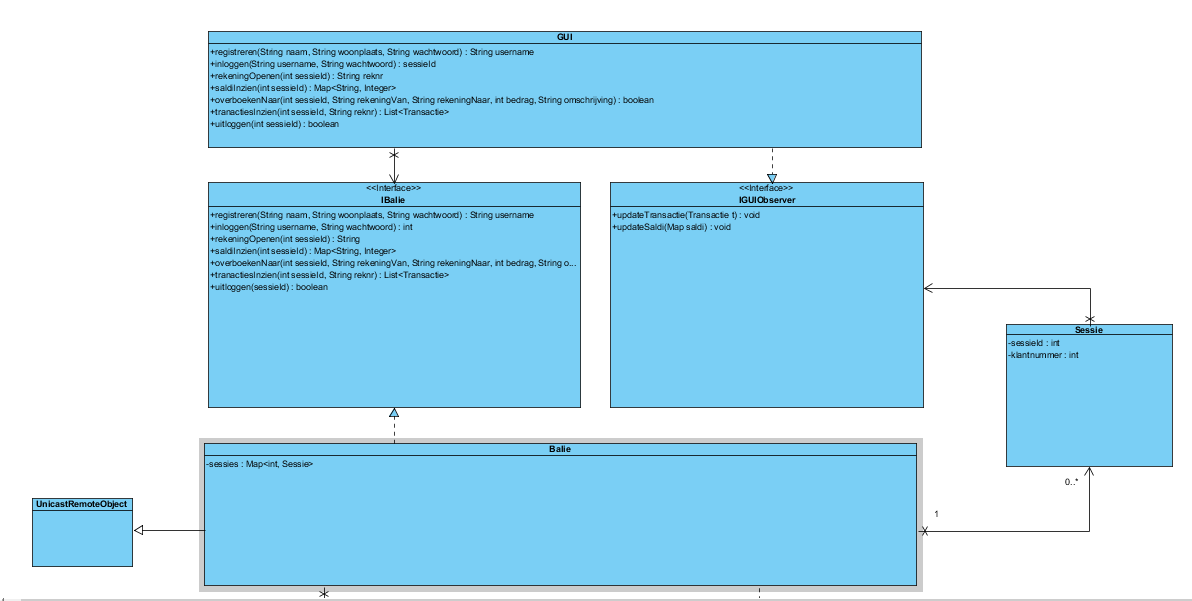
Voor het overboeken hebben we voor een andere methode gekozen. Een overboeking genereert altijd een transactie (mits voldoende saldo/kredietlimiet). Via de centrale bank kan later blijken dat storten op een tegenrekening niet lukt. De centrale bank zal dan een transactie terugsturen.

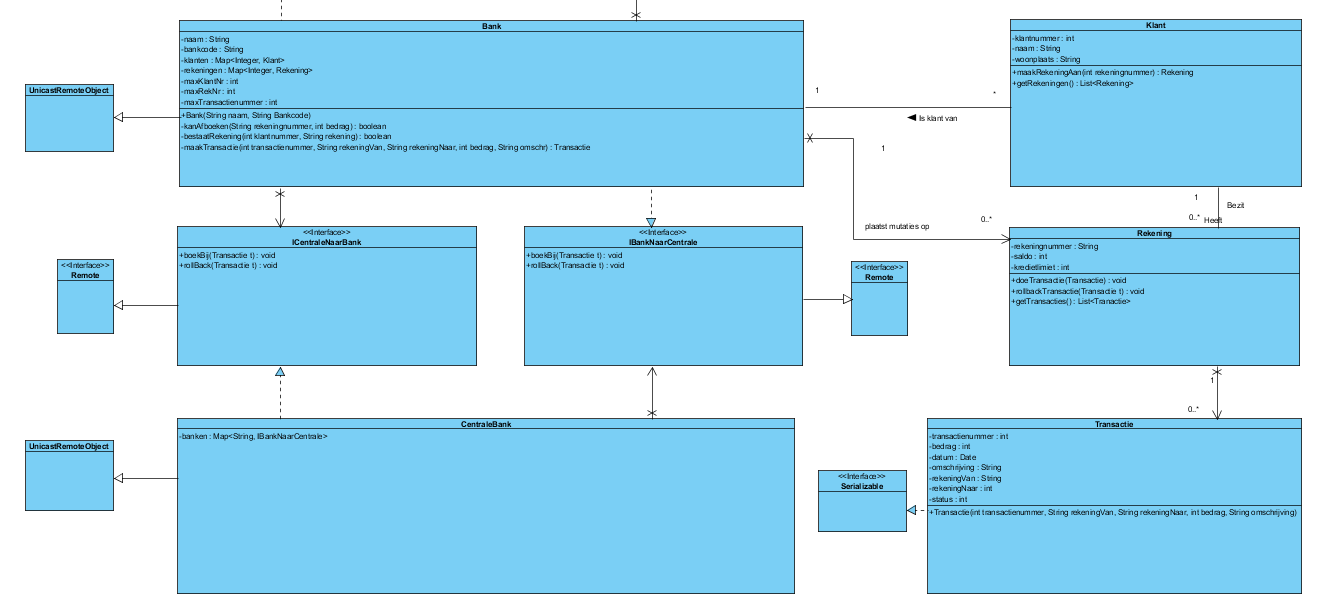
Een vereenvoudigde weergave is dan als volgt:



## Klassediagram v3

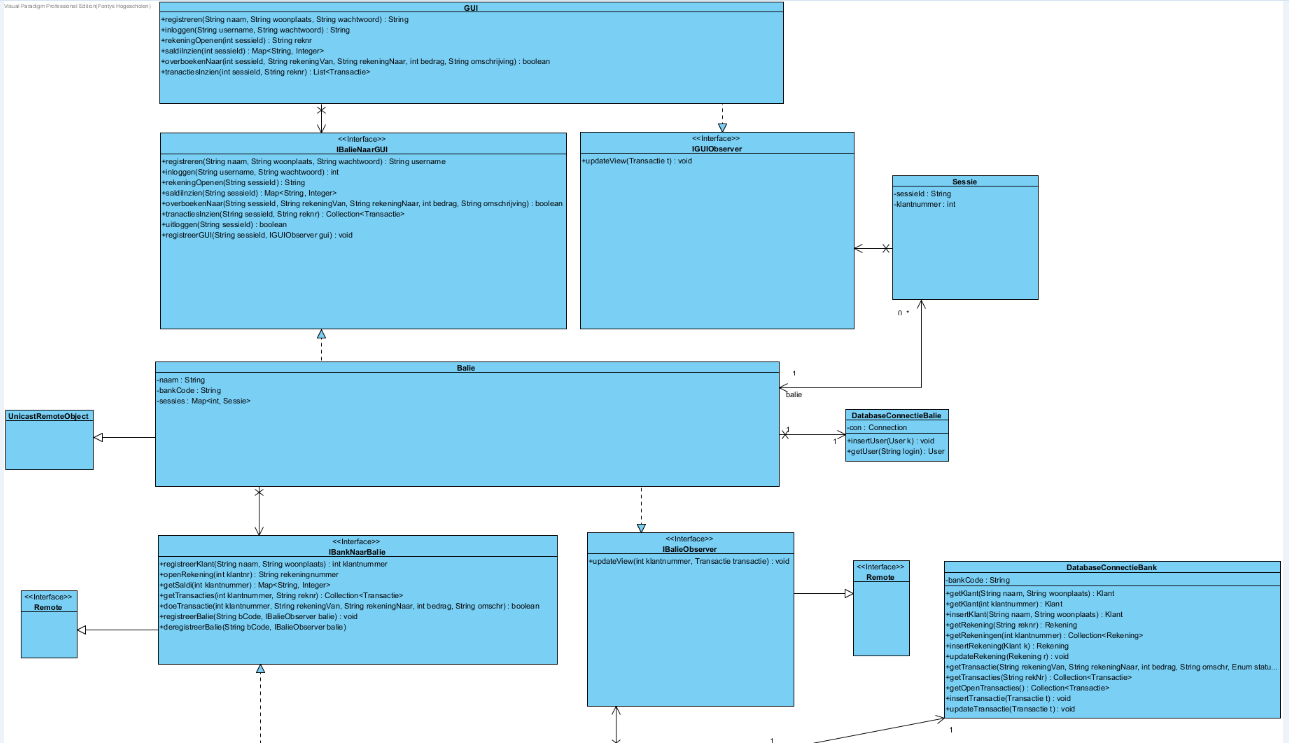
Bij voorgaande sequence diagrammen bleek het o.a. niet mogelijk om updates te pushen naar de client (vanuit de balie). Dat heeft er toe geleid dat het klassediagram weer grondig gewijzigd is. Hopelijk wordt nu ook beter zichtbaar wat de interfaces beschrijven, en waar deze nodig zijn of aangeboden worden.

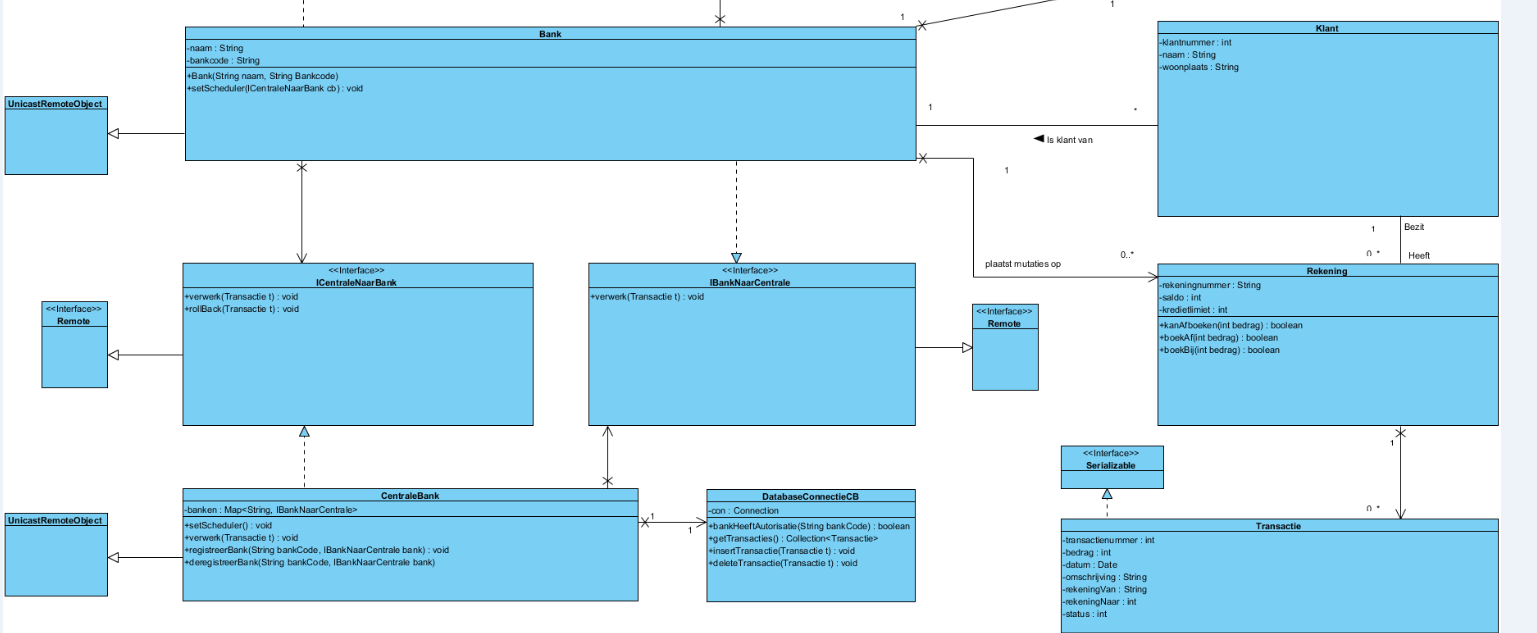




## Klassediagram v4 (definitief)

In het definitieve klassediagram is de klasse welke communicatie met de database verzorgt ook opgenomen,





# 5. Realisatie niet functionele eisen

**Beschikbaarheid**

* *Als een nieuwe bank zich registreert bij de centrale bank, zijn alle rekeningen van deze bank direct beschikbaar voor overboekingen.*
* *De centrale bank is maximaal 48 uur per jaar en 12 uur per etmaal niet beschikbaar.*
* *Elke bank is maximaal 168 uur per jaar en 16 aansluitend per dag niet beschikbaar.*

Gezien de korte tijdspanne besteden we hier geen aandacht aan. In de praktijk zouden we dit kunnen realiseren met het inrchten van meerdere servers voor centrale bank, bank en balie.

**Performance**

* *Overboekingen worden binnen 5 minuten verwerkt en zichtbaar gemaakt voor beide rekeninghouders.*
* *Elke bank kan service verlenen aan minimaal 1000 clienten tegelijk.*

Waar mogelijk gebruiken we RMI voor communicatie tussen gedistribueerde systemen. Tussen de GUI en Balie zal communicatie op een andere manier verlopen. Deze keuze zal nader onderzocht en onderbouwd worden waarbij de performance per methode ook meegewogen wordt.

**Persistentie**

* *Alle banken gebruiken een MySQL database. Banken zijn zelf verantwoordelijk voor backups van de database.*

Gegevens van de Balie, Bank en CentraleBank worden gepersisteerd m.b.v. een database. Alle gegevens welke niet sessie-uniek zijn zullen weggeschreven worden (Een sessieID is bijvoorbeeld sessie uniek en wordt niet weggeschreven).

**Veiligheid**

* *Al het webverkeer verloopt via het https protocol.*
* *Een sessie wordt na 5 minuten inactiviteit automatisch uitgelogd.*
* *Wachtwoorden worden versleuteld opgeslagen.*

Een mogelijke toevoeging voor veiligheid zou een extra authenticatie voor het versturen transacties zijn, deze mogelijkheid hebben we voor nu weggelaten i.v.m. beperkte tijd.

**Betrouwbaaheid**

Waar mogelijk (en nodig) is de code succesvol met Unit tests getest. In hoofdstuk 6: Testen staan alle tests uitgelegd.

**Usability**

* *De interface voor de rekeninghouders is gebruiksvriendelijk en intuïtief.*

De user interface zal getest worden met onervaren testgebruikers

# 6. Testen

Waar mogelijk zijn methodes getest met unit test. De geteste klassen/methodes staan hieronder beschreven

* Balie
  + Inloggen()
* Bank
  + registreerKlant()
  + openRekening()
  + doeTransactie()
  + getSaldi()
  + getTransacties()
* CentraleBank
  + registreerBank()

Het overige deel van de broncode is getest met een systeemtest waarin diverse gewenste en alternatieve flows zijn getest. Alle testen zijn hierbij geslaagd.

|  |  |  |
| --- | --- | --- |
| ID functionele eis | Naam | Omschrijving |
| F.1 | Registreren | Een klant kan zich registreren |
| F.2 | Sessie starten | Een klant kan een sessie starten (inloggen) |
| F.3 | Rekening Openen | Een klant kan een rekening openen |
| F.4 | Saldo inzien | Een klant kan het saldo van een van zijn rekeningen inzien |
| F.5 | Geld overmaken | Een klant kan geld overmaken van een van zijn rekeningen naar een andere rekening |
|  | Geld overmaken | Een klant boekt zonder succes geld over naar niet bestaande rekening |
|  | Geld overmaken | Een klant boekt zonder succes geld over naar niet bestaande bank |
| F.6 | Transacties inzien | Een klant kan de transacties van een van zijn rekeningen inzien |
| F.7 | Sessie beëindigen | Een klant kan zijn sessie beëindigen |

Tenslotte is het systeem en de usability van de GUI getest met een onervaren gebruiker in een systeem acceptatie test. De gebruiker is gevraagd om functionele eisen 1/tm7 (zie bovenstaand) uit te voeren. Dit is zonder problemen gelukt.

# 7. Niet gerealiseerde doelstellingen

Enkele eisen zijn niet binnen de beschikbaar gestelde tijd behaald, met gevolgen voor de kwaliteit van het systeem.

* De GUI wordt niet via een webpagina aangeboden aan de gebruiker, maar via een Java applicatie.
* De communicatie tussen GUI en balie verloopt niet via een SOAP protocol maar via Java’s RMI.
* De bank maakt gebruik van een ‘Mock’ database (en houdt daarmee gegevens niet langer dan 1 sessie vast) in plaats van gegevens naar een MySQL database weg te schrijven.
* Wachtwoorden worden niet versleuteld overgestuurd.
* De performance is niet duidelijk getest.
* De performance kan (waarschijnlijk) verbeterd worden door het slimmer communiceren met de database. Bijvoorbeeld door het meer ophalen/wegschrijven in batches en meer gegevens tijdelijk in het geheugen bewaren. Bij een reële bank applicatie zou een database pool voor het bijhouden en uitgeven van database-connecties een gewenst onderdeel zijn.