Architectuur document

Project: Internetbankieren

Project team: BankingSupport

Teamleden: Gijs Hendrickx (2389789)

Nadiv Tjong A Hung (2380382)

Opdrachtgever: De ING Bank

Versie: 0.1

Versiedatum: 19 november 2014

Status: Concept

# Documenthistorie

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Versie | Wijzigingen | Auteur | Datum | Verzendlijst | Verzoek |
| 0.1 | Opzet document | Nadiv Tjong A Hung | 19 november 2014 | Gijs Hendrickx | C |

Inhoud

Documenthistorie 2

H1 Inleiding 4

H2 Domeinmodel 5

H3 Persistentie 6

H4 Communicatie 7

H5 Realisatie niet-functionele eisen 8

H6 Componenten 9

H7 Deployment 11

H8 Specificatie van interfaces 12

# H1 Inleiding

**Context**

Een van de hoofddiensten die de Nederlandse banken aanbieden aan hun klanten betreft het overmaken van geldbedragen naar anderen. Door het sluiten van een transactie kan één rekeninghouder een geldbedrag overboeken naar de rekening van een ander. Het is gebruiksvriendelijk en gemakkelijk om dit elektronisch te kunnen doen. De ING Bank wil voor deze functionaliteit een nieuwe applicatie hebben. Het gaat daarbij om één bank, met meerdere accounts. De accounts kunnen geld over boeken naar andere accounts van dezelfde bank, of van een andere bank. Voor het overboeken van geld naar een andere account, is er een overboekcentrale die contact heeft met alle banken.

**Applicatie**

De internetbankieren applicatie zal bestaan uit 3 onderdelen: een Client, een Bank en een Centrale. De onderdelen worden via internet met elkaar verbonden. De Client kan zich identificeren, transacties uitvoeren, een nieuwe account aanmaken, kredietlimiet aanpassen, en een sessie beëindigen.

**Niet-functionele eisen**

Onderstaande niet-functionele eisen uit het user requirements document (URS v1.1) worden besproken in dit document:

Beveiliging:

Accounts worden geverifieerd op correctheid van inlogcode en wachtwoord (URS Q.1). Transacties mogen maximaal 10 minuten duren om de integriteit van een transactie te beschermen (URS Q.2).

Gebruiksvriendelijkheid:

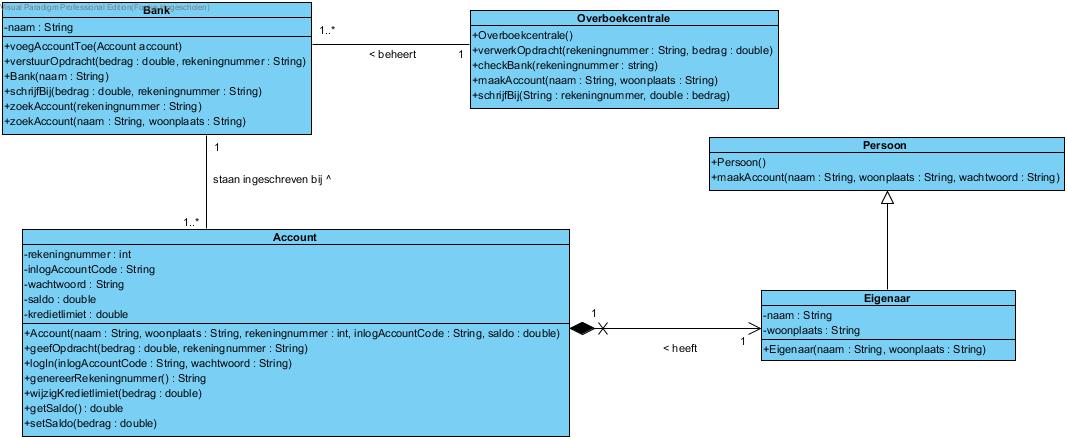
De voortgang van een transactie (resterende tijd) wordt getoond aan de gebruiker (URS Q.3).

**Doel van dit document**

In dit document wordt de architectuur van de internetbankieren applicatie verantwoord. Daarnaast worden de interfaces besproken voor de communicatie tussen verschillende componenten. Er worden zeven aspecten van de architectuur besproken: domeinmodel, persistentie, communicatie, realisatie van niet-functionele eisen, componenten, deployment en specificatie van interfaces.

# H2 Domeinmodel

**Klassendiagram**



Figuur 1: Klassendiagram van het domein.

In figuur 1 is het klassendiagram van het domein te bekijken. Hierin zijn de volgende klassen te onderscheiden:

* Account: Representeert een account van een klant. Hier valt de rekening van de account onder, maar ook de inloggegevens. Een Account heeft een rekeningnummer, een inlogAccountCode, een wachtwoord, een saldo en een kredietlimiet . Via een account kan worden ingelogd, een transactie gepleegd, het kredietlimiet worden gewijzigd,
* Eigenaar: Representeert een rekeninghouder. Wordt geïdentificeerd door een combinatie van naam en woonplaats. Elke Eigenaar heeft een account.
* Bank: Representeert een bank. Een bank heeft een naam, en verschillende Accounts. De bank regelt in principe alle zaken gerelateerd aan bankieren voor klanten, maar bij sommige situaties is het nodig om contact te hebben met de overboekcentrale (bijvoorbeeld bij een transactie naar een rekening van een andere bank).
* Overboekcentrale: Representeert een overboekcentrale. Dit is een centraal orgaan dat een lijst van banken bijhoudt. Als er een transactie tussen rekeningen van twee banken moet plaatsvinden, kan de centrale als middelpunt tussen de twee banken fungeren.

**Afbakening**

Dit domeinmodel representeert een vereenvoudigde bankierensituatie. Onderwerpen zoals beveiliging en logistiek worden slechts in beperkte mate aangekaart in dit document. De interfaces die worden besproken betreffen de facilitatie van transacties en account-gerelateerde zaken zoals het wijzigen van een kredietlimiet.

# H3 Persistentie

**Eigenschappen van objecten m.b.t. persistentie**

Om het bankieren goed te laten verlopen, moeten de volgende eigenschappen van objecten worden opgeslagen:

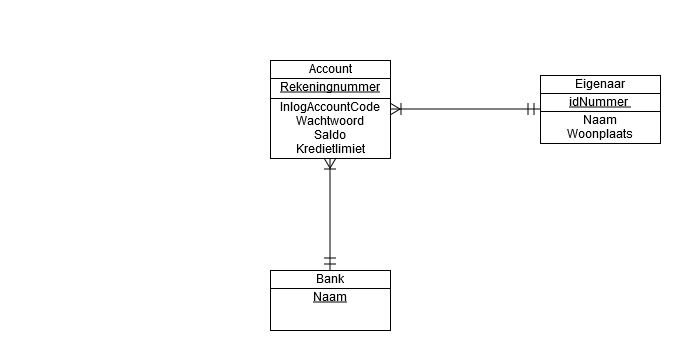
* Account: rekeningnummer, inlogAccountCode, wachtwoord, saldo en kredietlimiet.
* Eigenaar: naam en woonplaats.
* Bank: naam.

**Data-opslag voor bankieren**

Gegevens van accounts en eigenaren worden opgeslagen in een database. Voor deze oplossing is gekozen omdat er waarschijnlijk een groot aantal accounts voor een groot aantal cliënten zullen zijn. Het is dan niet praktisch om via serialisatie te werken, een database is voor dergelijke informatieopslag handiger.

Vanuit het oogpunt van veiligheid is het natuurlijk ook veel beter om gevoelige account- of gebruikersinformatie in een database te op te slaan, in plaats van bijvoorbeeld op een bestand op de laptop van een gebruiker.

**ERD**



# H4 Communicatie

**Eigenschappen van objecten m.b.t. communicatie**

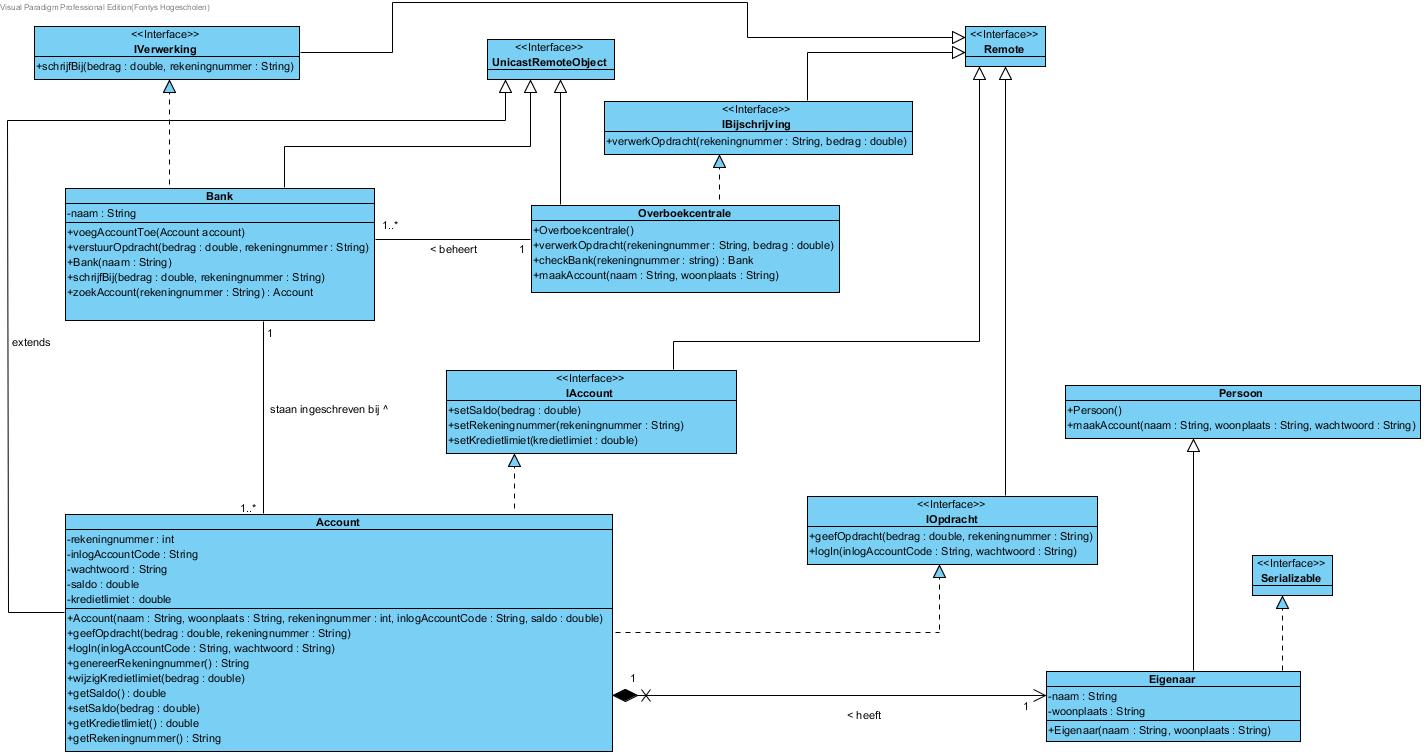
Bij het inlogproces van een gebruiker worden de volgende eigenschappen van objecten gecommuniceerd:

* Account: inlogAccountCode, wachtwoord, saldo, kredietlimiet, rekeningnummer.

Daarnaast wordt bij een transactie het rekeningnummer van de versturende en ontvangende rekening gecommuniceerd, en het over te maken bedrag. Bij een kredietlimietwijziging wordt het nieuwe kredietlimiet gecommuniceerd.

**Dataverkeer voor inloggen, kredietlimiet wijzigen en transacties**

Voor alle communicatie is gekozen voor gebruik van Remote Method Invocation (RMI). Dit zorgt ervoor dat dataverkeer snel en betrouwbaar kan verlopen. Voor inloggen wordt het pull-principe gebruikt en voor transacties en kredietlimiet aanpassen het push-principe. Daarnaast gebruiken wij voor de realisatie van   
kwaliteitsattributen het push-principe: dit geldt bij het limiteren van de tijd van een transactie tot maximaal 10 minuten (URS Q.2) als bij het weergeven van de voortgang van een transactie aan een cliënt (URS Q.3).

Figuur 2: klassendiagram voor communicatie m.b.v. RMI.

Het klassendiagram in figuur 2 representeert de relaties tussen betrokken klasses en interfaces t.b.v. communicatie volgens het push- en pullprincipe.

De trekkende component Eigenaar is hierbij bij het inloggen leidend (zie H6). Na aanroep van de methode login() uit de IOpdracht interface worden de accountgegevens opgehaald van de BankSysteem component en gestuurd naar de betreffende Gebruiker.

Bij het wijzigen van kredietlimiet is tevens de Gebruiker leidend, er wordt via de setKredietlimiet() methode van de IAccount interface een nieuw limiet verstuurd naar het BankSysteem (er vindt hier dus een push plaats).

Ook bij het maken van een transactie leidt de Gebruiker. Via de geefOpdracht() methode van de IOpdracht interface wordt een opdracht tot het doen van een transactie gestuurd naar het betreffende BankSysteem. Vanaf daar wordt de schrijfBij() methode van de IVerwerking interface gebruikt om geld over te maken naar een andere rekening. Als het gaat om een rekening van een andere bank, wordt er eerst bij de Centrale component de verwerkOpdracht() methode van interface IBijschrijving aangeroepen alvorens het geld bij de andere rekening wordt overgeboekt.

Bij het sturen van actieve sessietijd is de Gebruiker leidend, terwijl bij het versturen van transactiestatus-updates het BankSysteem leidt.

# H5 Realisatie niet-functionele eisen

**Betrouwbaarheid**

Ter verzekering van de betrouwbaarheid van het systeem, worden er checks gedaan op de combinatie van inlogAccountCode en wachtwoord. Dit is uiteraard ook gerelateerd aan de verzekering van veiligheid. Daarnaast mogen transacties maar maximaal tien minuten duren, wederom ten behoeve van betrouwbaarheid en veiligheid.

**Gebruikersvriendelijkheid**

Bij het plegen van een transactie, wordt steeds de huidige status van de transactie weergegeven aan de Gebruiker. Op deze manier kan de gebruiker ten alle tijden zien wat er gebeurd met zijn of haar geld.

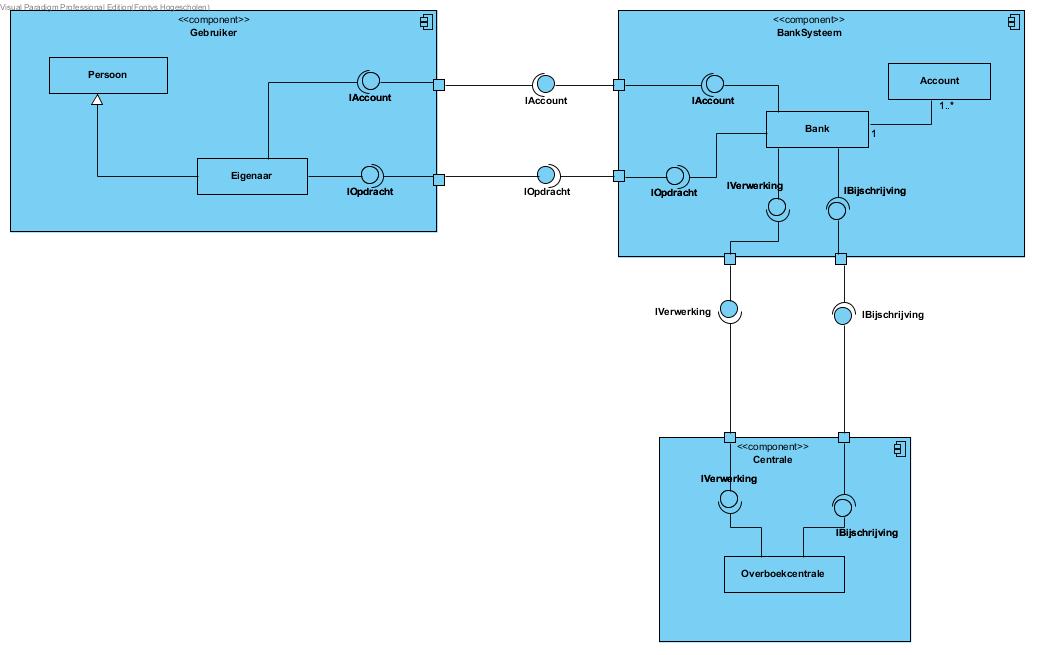
**Beveiliging**

Zie de uitleg bij het kopje Betrouwbaarheid.

**Schaalbaarheid**

Ten behoeve van schaalbaarheid maken wij gebruik van een database voor persistentie. Ons databaseontwerp is simpel maar flexibel. Op deze manier kan zowel een groot als klein aantal klanten worden gedekt.

# H6 Componenten

**Componentendiagram**

Figuur 3: Componentendiagram.

**Koppeling tussen componenten**

Omdat er gebruik wordt gemaakt van Remote Method Invocation (RMI), is een onderverdeling in componenten noodzakelijk. De interfaces die worden gebruikt als communicatiemiddel tussen deze componenten zijn IAccount, IOpdracht en IBijschrijving en IVerwerking.

**Synchronisatie tussen componenten**

De componenten kunnen niet communiceren totdat er een account actief is (ofwel als de interface IAccount is ontvangen door de eigeneer)

**Services per component**

IAccount: het overgeven van accountgegevens van de bank aan de client, die de gegevens verifieert en een accountstubobject realiseert.

IOpdracht: het geven van een transactieopdracht of een inlogactie van de client naar de bank.

IBijschrijving: een signaal van de bank naar de centrale dat er een bijschrijving op een andere rekening moet plaatsvinden.

IVerwerking: antwoord van de centrale op bovenstaande interface, geeft een signaal dat de transactie voltooid is.

Services die per component worden aangeboden:

* Account: IAccount (accountgegevens), IOpdracht (verzoek tot opdracht of account)
* Bank: IBijschrijving
* Overboekcentrale: IVerwerking

**Allocatie van objecten per component**

* Account: instanties van accounts die ingelogd zijn
* Bank: instanties van benodigde banken voor transactie
* Overboekcentrale: instantie van (enkele) overboekcentrale

**Remote objecten**

* Account
* Bank
* Overboekcentrale

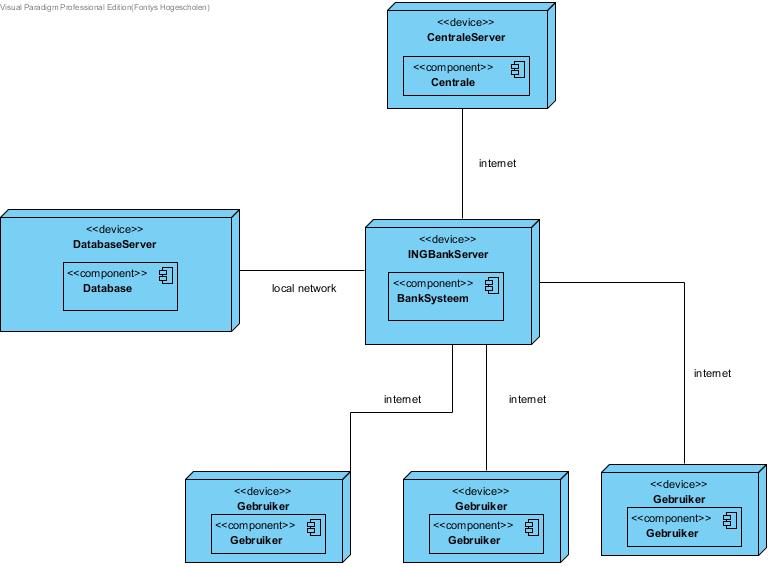
**Packagestructuur**

Package Generic: Account, Eigenaar, Persoon, Overboekcentrale, Bank

Package Interfaces: IAccount, IOpdracht, IVerwerking, IBijschrijving

# H7 Deployment

**Deploymentdiagram**



Figuur 4: Deploymentdiagram.

Deployment van de bankierenapplicatie is afgebeeld in figuur 3 middels een deploymentdiagram. Het bestaat uit de volgende onderdelen:

* CentraleServer: de overboekcentrale van de Nederlandse Bank (locatie: Amsterdam).
* INGBankServer: de server van de ING Bank waarop de applicatie draait (locatie: Amsterdam).
* DatabaseServer: de database server waarop de database van de bank draait, verbonden via een lokaalnetwerk met de applicatieserver (locatie: Amsterdam).
* Gebruiker: verschillende gebruikers, die de bankierenapplicatie lokaal draaien en verbonden met de applicatieserver zijn via internet.

# H8 Specificatie van interfaces

**Interface IAccount**

Methode: setSaldo

Preconditie: Account bestaat, verbinding actief

Beschrijving: Haalt het saldo van de ingevoerde account op

Exceptie: Als account niet bestaat, of als er geen verbinding is

Methode: setRekeningnummer

Preconditie: Account bestaat, verbinding actief

Beschrijving: Haalt het rekeningnummer van de ingevoerde account op

Exceptie: Als de account niet bestaat, of als er geen verbinding is

Methode: setKredietlimiet

Preconditie: Account bestaat, verbinding actief

Beschrijving: Haalt het kredietlimiet van de ingevoerde account op

Exceptie: Als account niet bestaat, of als er geen verbinding is

**Interface IOpdracht**

Methode: geefOpdracht

Preconditie: Verbinding actief, saldo > 0, ingelogd

Beschrijving: Verzoek aan bank om transactie te starten

Exceptie: Tegenaccount bestaat niet, saldo <= 0, geen verbinding

Methode: logIn

Preconditie: Nog niet ingelogd, verbinding actief

Beschrijving: Verzoek aan bank om in te loggen

Exceptie: Geen verbinding

**Interface IVerwerking**

Methode: schrijfBij

Preconditie: Bank heeft verbinding met centrale

Beschrijving: Bank vraagt aan centrale om bijschrijving bij account

Exceptie: Geen verbinding

**Interface IBijschrijving**

Methode: verwerkOpdracht

Precondities: Centrale heeft verbinding met desbetreffende bank

Beschrijving: De transactie wordt voltooid, het saldo wordt bijgeschreven op tegenaccount

Exceptie: Geen verbinding