Bankieren Architectuurdocument

By

Koen van Vliet – Thomas van Dongen

GSO

Marcel Meesters

|  |  |  |
| --- | --- | --- |
| Versie | Datum | Bewerking door |
| 0.1 | 11/28 | Koen van Vliet |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[Inleiding 3](#_Toc468796480)

[Context 3](#_Toc468796481)

[Applicatie 3](#_Toc468796482)

[Niet-functionele eisen 4](#_Toc468796483)

[Domeinmodel 5](#_Toc468796484)

# Inleiding

Dit document, het architectuurdocument, geeft richting aan de ontwikkelaars van dit project. Het schetst een beeld van het doel van het project. Ook helpt het de functionele en niet-functionele eisen daadwerkelijk te realiseren door deze goed vast te leggen. Het document doelt er ook op bepaalde risico’s van tevoren te verkennen om deze in te perken. Daarnaast word het project hier ook gedocumenteerd om wijzigingen vast te leggen, zodat er geen vergissingen worden gemaakt in de implementatie.

## Context

Bankieren via het internet wordt steeds gebruikelijker, zo niet onvermijdelijk. Personen kunnen thuis inloggen op een applicatie waarin ze hun bankrekening kunnen inzien en, indien hun saldo dat toestaat, geld overschrijven naar een andere rekening. Dit gebeurt niet van applicatie tot applicatie, maar via een centrale server die alle transacties beheert. Deze overboekingen zijn mogelijk binnen dezelfde bank, maar ook naar rekeningen van andere banken. Een rekening wordt beheerd door één persoon maar mag wel op meerdere locaties tegelijk worden ingezien.

## Applicatie

Het gedistribueerd bankierensysteem bestaat uit verschillende onderdelen. Er is een applicatie voor de rekeninghouders, zodat deze hun rekening in kunnen zien, en een applicatie voor de bank die de rekeningen van zijn eigen rekeninghouders overziet. Daarnaast is er een centrale transactie-applicatie die de transacties tussen verschillende banken overziet.

De applicatie voor de rekeninghouders is erg simpel; Deze communiceert met de centrale bank waarbij de rekeninghouder zijn rekening heeft geopend. De rekeninghouder kan daar zijn rekeningsaldo opvragen of een transactie aanmaken en versturen.

De applicatie voor de banken overziet de rekeningen van hun rekeninghouders, de transacties bínnen diezelfde bank en indien er een transactie wordt verzonden naar een andere rekening stuurt de bank deze door naar de centrale transactie-applicatie.

Deze centrale transactie-applicatie ontvangt de eerdergenoemde transacties en verwijst ze door naar de juiste bank.

## Niet-functionele eisen

**Actueel:**

* De applicatie update de saldo’s realtime zodat de gebruiker, wanneer mogelijk, zijn juiste saldo ziet.
* Transacties worden direct doorgevoerd naar de gebruiker zodat ook na transacties van de rekeninghouder zelf of van andere rekeninghouders het saldo wordt bijgewerkt.

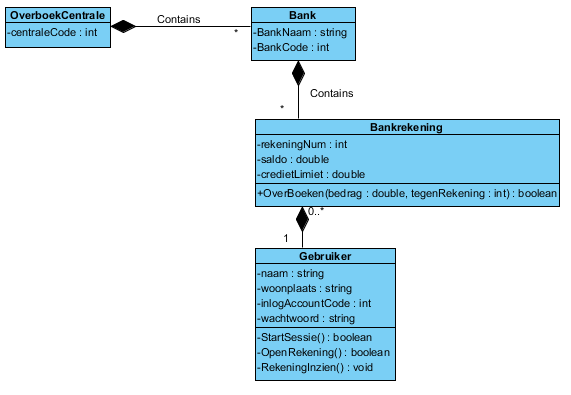
**Performance:**

* Alhoewel een bankierapplicatie naast realtime updates niet per-se bijzonder gefocussed hoeft te zijn op de performance, moet het natuurlijk niet een hele computer laten vastlopen. De applicatie dient soepel te moeten draaien en de GUI moet niet lang vastlopen.

**Schaalbaarheid:**

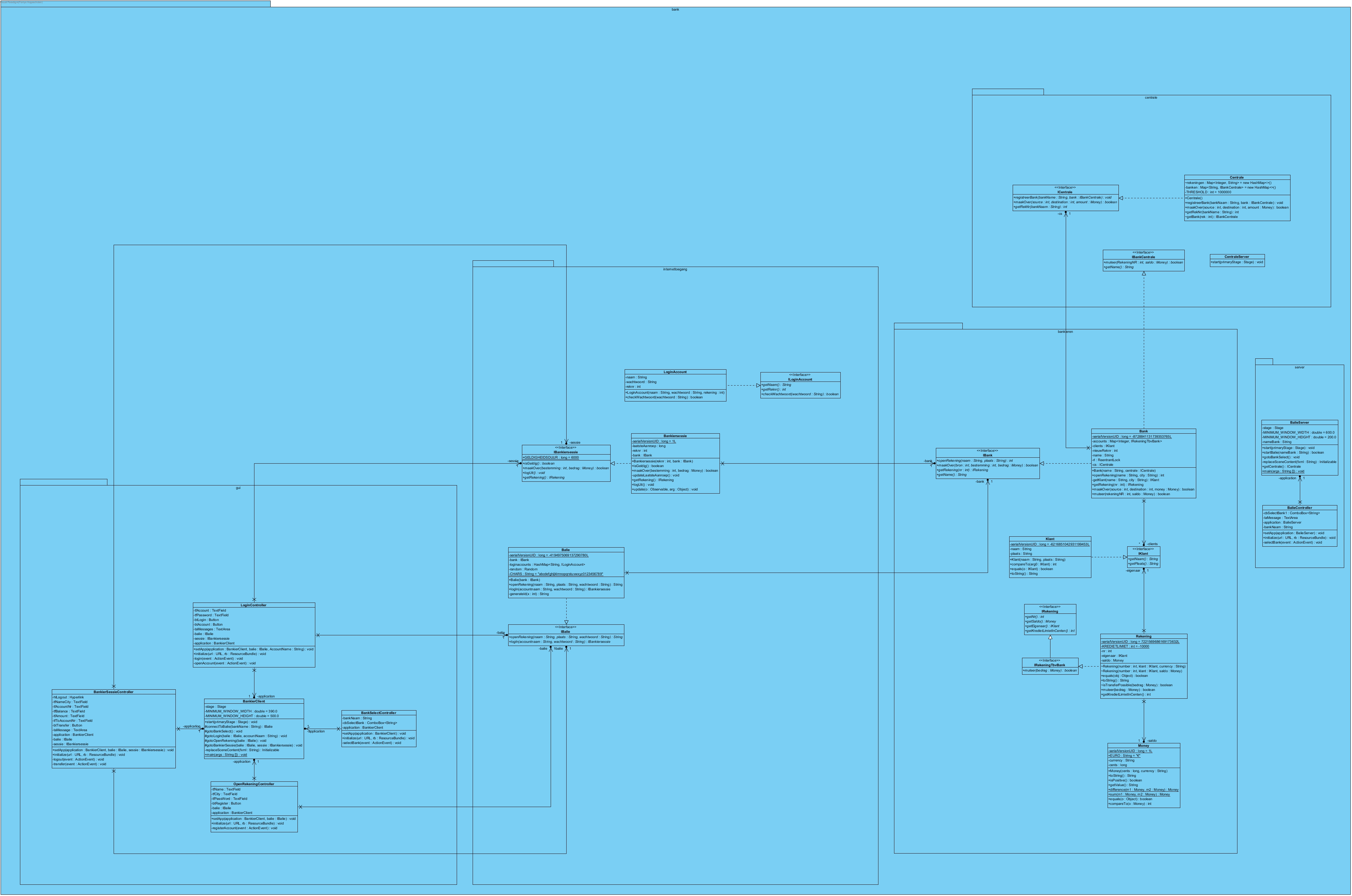
* De applicatie moet uitbreidbaar kunnen zijn voor meerdere banken, ook moeten er nieuwe gebruikers en rekeningen aangemaakt kunnen worden met virtueel geen limiet (de servers en computers zijn de grens).

# Domeinmodel

Het eerste domeinmodel. In dit model verkenden we de applicatie en bedachten hoe de kern van de bank in elkaar zou zitten.

De gebruiker zou een applicatie hebben die verbinding maakt met een andere applicatie die draait bij de bank. Om ook overboekingen mogelijk te maken naar andere banken, is er een 3e applicatie die draait bij een centrale server. Deze applicatie handelt alle inter-bank transacties af.

Om tijdens het programmeren een overzicht te hebben van het volledige project is het volgende klassendiagram gemaakt. ([Bijlage](Class%20Diagram1.jpg), [VP project](vpInternetBankieren.vpp)).



De centrale houdt de rekeningen bij, en banken registreren zich zodra ze verbinden met de centrale aan bij deze centrale. Zo kan een bank een nieuw rekeningnummer opvragen en de centrale weten naar welke bank hij moet gaan voor een bepaalde rekening. Zodra een bank een overboeking maakt, gaat deze via de centrale, die vervolgens aan de juiste bank vraagt om het geld over te maken.

Er is een Observer Pattern toegepast op het overmaken van geld. Zodra een rekening wordt aangepast, wordt de bijbehorende view geupdate met het nieuwe saldo van de rekening. Dit gaat op een soortgelijke manier als de ingebouwde Java observer, maar is meer geïntegreerd met de rekening-klasse. Deze keuze hebben we gemaakt omdat een Java object niet meerdere parent-klassen kan overerven. De rekeningklasse is al serialisable en kon niet direct een Observable klasse overerven. In plaats van een nieuw ‘tussenobject’ te programmeren implementeert de rekening-klasse een ObservableRekening interface.

De gebruikerklasse houdt de informatie van klanten bij. Gebruikers kunnen meerdere rekeningen openen, en gebruikers kunnen deze vanaf meerdere punten tegelijk inzien.

De rekeningklasse houdt het saldo van de gebruiker bij, en houdt ook het saldo bij, doet de overboekingen en controleert of deze mogelijk zijn.

De bank overziet al zijn klanten en rekeningen, en communiceert met de bankcentrale voor externe boekingen.

De centrale handelt externe boekingen af door met de juiste banken te communiceren.

# Componenten

# Persistentie

De applicatie bevat informatie. Deze informatie moet opgeslagen worden. Om deze informatie op te slaan hebben we meerdere keuzes. Wij willen de data opslaan met behulp van serialization. Deze methode is misschien niet het meest prettig met betrekking tot handelbaarheid van informatie, maar het is wel performance-efficient, makkelijk te implementeren en veiliger dan een externe database.

#### Realtime en persistente informatie

De informatie die we realtime willen bijhouden, zoals sessies en overboekingen, worden direct in het programma afgehandelt. Informatie die moet worden opgeslagen voor lange tijd, zoals saldos en rekeningen, worden via serialization in flat-files opgeslagen op de server van de bank.

#### Type informatie

Deze informatie bevat de gebruikers en hun **gegevens**, de **rekeningen** van deze accounts en de **informatie** die daarbij hoort en de **banken** zelf. De centrale zal ook bijhouden welke **rekeningen** er zijn aangemaakt.

# Communicatie

Er zijn meerdere applicaties die op meerdere locaties worden uitgevoerd. Deze applicaties moeten dus met elkaar communiceren via het internet. De informatie die wordt verzonden zijn de **rekeningnummers, geregistreerde banken, rekeningen** en **overboekingen**.

#### Type Communicatie

Wij maken gebruik van RMI (Remote Method Invocation) zodat we de applicaties op een prettige manier realtime met elkaar kunnen laten communiceren. De gebruiker communiceert met de bank, en de bank communiceert indien nodig met de centrale. Er wordt gebruik gemaakt van het push principe voor een transactie, en een pull principe voor het bijhouden van saldos. De gebruiker kan ook rekeningen aanmaken en inzien bij een bank (respectievelijk push en pull).

#### Geld Overboeken

Zodra een gebruiker een overboeking maakt, zal deze eerst naar de bank gaan. De bank verwijst deze door naar de centrale die, na te vragen of de overboeking mogelijk is, de overboeking doorstuurt naar de desbetreffende bank. Vervolgens handelt de bank deze overboeking af en laat de bijbehorende sessie weten dat er een update nodig is voor de geopende rekening.

#### Rekening aanmaken

Gebruikers kunnen een rekening aanmaken. Zodra ze zich registreren wordt er aan via de bank aan de centrale gevraagd welke rekeningnummers nog vrij zijn. De bank maakt met dit nummer een nieuwe rekening aan.

# Realisatie niet-functionele eisen

De informatie moet betrouwbaar zijn. Bij het overboeken van geld maken wij gebruik van checks die kijken of de overboeking wel mogelijk is. Als deze check faalt, wordt de overboeking niet voltooid en het afgeschreven geld wordt teruggedraaid. Ook bij het aanmaken van rekeningen wordt op de centrale gekeken of de rekening aangemaakt kan worden. Deze checks zorgen ervoor dat er geen inconsistentie onstaat in de rekeningen en hun saldos.

Voor de schaalbaarheid ontwikkelen we de applicatie zo dat nieuwe banken makkelijk kunnen worden toegevoegd, en deze worden automatisch geregistreerd bij de centrale zodra ze zich daar melden. Ook kunnen er meerdere gebruikers en rekeningen aangemaakt worden.

# Deployment