Technisch ontwerp

IPBDAM5

**Project: IPBDAM5**

**Studiejaar: 2020/2021**

**Periode: 4**

**Groep: 7**

**Projectbegeleider/Scrummaster: Roland Westveer**

**Product Owner: Rudolf Brugman**

**Projectleden:**

* **Jelle van den Bosch, 1121244**
* **Brain Walters, 1121120**
* **Edwin van Leeuwen, 1135325**
* **Damian Alkemade, 1121187**

# Versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Datum | Verandering | Auteur |
| 0.1 | 9-4-2021 | Document aangemaakt | Jelle van den Bosch |
| 0.2 | 14-5-2021 | Hoofdstukken aangemaakt en gevuld | Jelle van den Bosch |
| 0.3 | 18-5-2021 | Details aangepast | Jelle van den Bosch |
| 0.4 | 24-5-2021 | Kopjes aangevuld en aangepast | Jelle van den Bosch |
| 1.0 | 25-6-2021 | Laatste wijzigingen | Jelle van den Bosch |

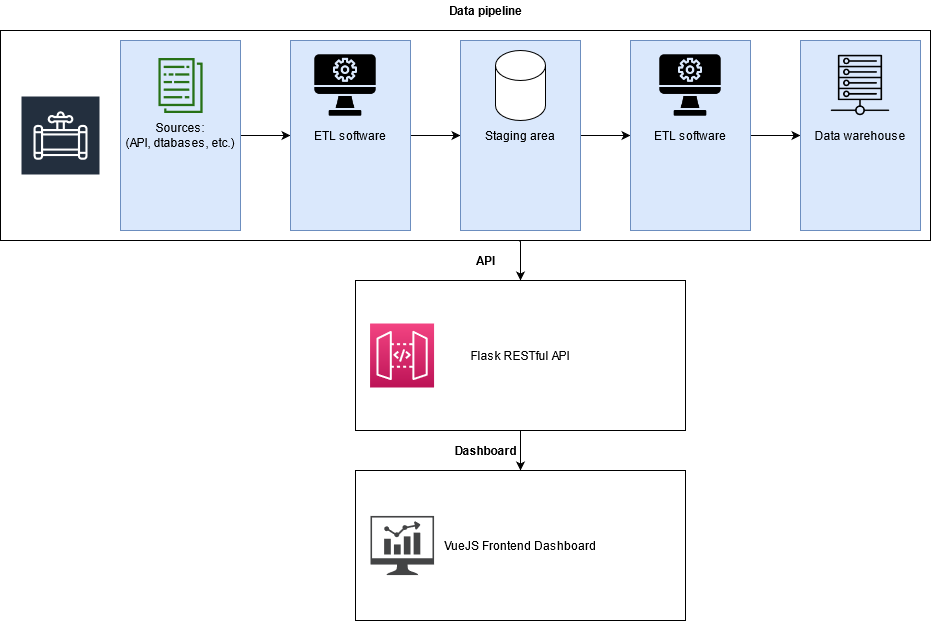
# Inhoudsopgave

Inhoud

[Versiebeheer 2](#_Toc75527540)

[Inhoudsopgave 3](#_Toc75527541)

# 1.0. Samenhang software componenten



**Toelichting:**

Elk van de drie hoofdcomponenten is een apart stuk software die middels MySQL en API requests met elkaar in verbinding staan.

*Data pipeline*

De hoofdtaak van de data pipeline is om het data warehouse te vullen met data afkomstig uit verschillende data bronnen en data die is ontstaan tijdens het ETL proces. Als eerst wordt data uit meerdere bronnen gehaald - denk hierbij aan Reddit, Twitter en Google – door het ETL proces, dan wordt het vervolgens getransformeerd naar het gewenste formaat om het dan in de staging database te zetten. Tot slot wordt de data middels een apart ETL uit de staging area gehaald en in het data warehouse opgeslagen.

*API*

Het doel van de API is om een tussenschakel te zijn tussen het dashboard en de database. De keuze hiervoor is om een aantal redenen:

* De frontend moet nooit direct interacteren met een database
* Scheidt de taken in twee, ontwerpprincipe
* Werkt gestructureerder

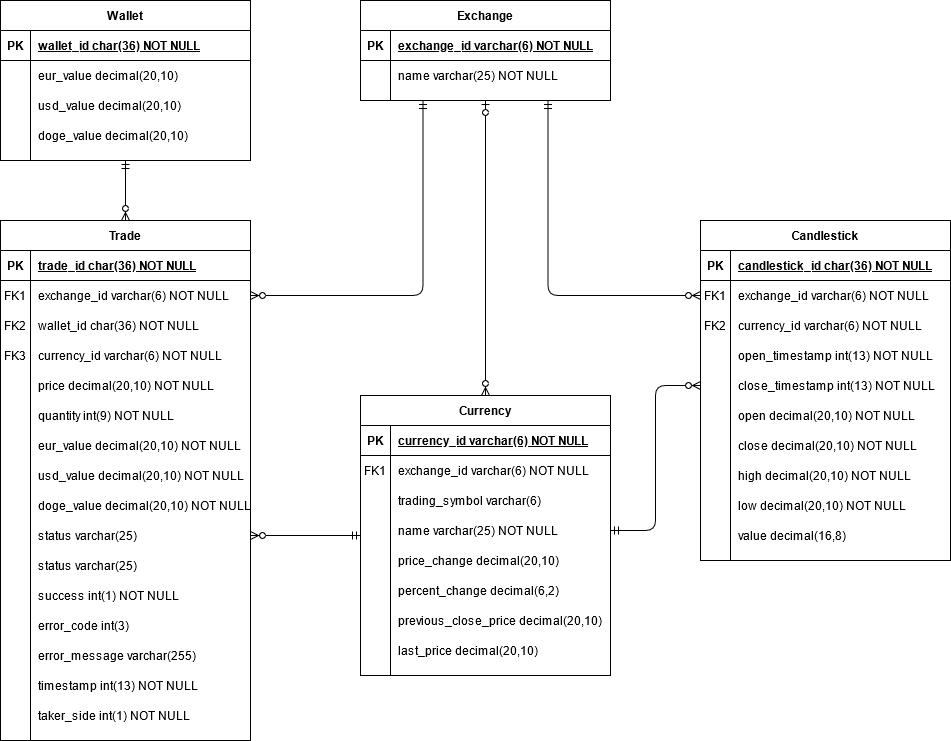
De API is gemaakt met Python en de Flask library volgens RESTful principes.

*Dashboard*

Het dashboard is hetgene dat uiteindelijk de grafieken toont en inzichten in de data geeft. Deze frontend is ontwikkeld middels het javascript framework VueJS. Het dashboard roept functies in de API aan die vervolgens de data uit het data warehouse als response terugsturen.

# 2.0 Technisch datamodel

**Staging database TDM**



**Toelichting**

**Candlestick tabel**

De *Candlestick* table bevat alle candlesticks per uur, per dag. Elke rij in de tabel is één candlestick.

*Relaties*

* De *Candlestick* tabel heeft een één op meer (1:N) relatie met de *Exchange* tabel. Elke candlestick komt **altijd** van één exchange, terwijl een exchange meerdere candlesticks **kan** hebben.
* De *Candlestick* tabel heeft een één op meer (1:N) relatie met de *Currency* tabel. Elke candlestick gaat **altijd** over een bepaalde currency, terwijl een currency meerdere candlesticks **kan** hebben.

*Sleutels*

* Elke *Candlestick* rij **moet** een foreing key bevatten die wijst naar een bepaalde *Exchange*.
* Elke *Candlestick* rij **moet** een foreing key bevatten die wijst naar een bepaalde *Currency*.

*Columns*

* *candlestick\_id:* Een unieke surrogate key die wordt gebruikt om elke candlestick te kunnen identificeren
* *exchange\_id:* Een foreign key die wijst naar een uniek identificeerbare rij in de *Exchange* tabel
* *currency\_id:* Een foreign key die wijst naar een uniek identificeerbare rij in de *Currency* tabel
* *open\_timestamp:* Het tijdsmoment dat het begin van een meetperiode aangeeft. Dit is altijd rond het begin van een nieuw uur, zoals: 2-2-2000 09:00:00 of 2-2-2019 14:00:36.
  + Genoteerd als Epoch timestamp in millisecondes.
  + Data type is een Integer
  + Lengte is 13
  + Voorbeeld: 1615550400000
* *close\_timestamp:* Het tijdsmoment dat het einde van een meetperiode aangeeft. Dit is altijd rond het einde van een uur, zoals: 2-2-2000 09:59:59 of 2-2-2019 14:59:36.
  + Genoteerd als Epoch timestamp in millisecondes.
  + Data type is een *Integer*
  + Lengte is 13
  + Voorbeeld: 1615550400000
* *open:* De waarde van de Dogecoin koers (prijs) op het moment dat de meetperiode (*open\_timestamp*) begon.
  + Genoteerd in ‘quote’ currency (EUR)
  + Data type is een *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 5,43434
* *close:* De waarde van de Dogecoin koers (prijs) op het moment dat de meetperiode (*close\_timestamp*) eindigde.
  + Genoteerd in ‘quote’ currency (EUR)
  + Data type is een *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 5,43434
* *high:* De hoogste waarde dat de Dogecoin koers (prijs) bereikte gedurende de meetperiode.
  + Genoteerd in ‘quote’ currency (EUR)
  + Data type is een *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 5,43434
* *low:* De laagste waarde dat de Dogecoin koers (prijs) bereikte gedurende de meetperiode.
  + Genoteerd in ‘quote’ currency (EUR)
  + Data type is een *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 5,43434
* *value:* De kwantiteit gekochte of verkochte Dogecoin.
  + Genoteerd in ‘base’ currency
  + Data type is *Decimal*
  + Lengte is 16,8
  + Voorbeeld: 50384662
* *amount\_of\_trades:* Aantal keren gedurende de meetperiode dat er werd gehandeld in Dogecoin.
  + Genoteerd in aantallen
  + Data type is *Integer*
  + Lengte is 8
  + Voorbeeld: 523

**Currency tabel**

De *Currency* table bevat actuele informatie over een bepaalde currency.

*Relaties*

* De *Currency* tabel heeft een één op meer (1:N) relatie met de *Candlestick* tabel. Elke candlestick gaat **altijd** over een bepaalde currency, terwijl een currency meerdere candlesticks **kan** hebben.
* De *Currency* tabel heeft een één op meer (1:N) relatie met de *Exchange* tabel. Elke currency informatie komt **meestal** van een bepaalde exchange.
* De *Currency* tabel heeft een één op meer (1:N) relatie met de *Trade* tabel. Elke trade wordt **altijd** gedaan in een bepaalde currency, terwijl een currency meerdere trades kan hebben.

*Sleutels*

* Elke *Currency* rij **moet** een foreign key bevatten die wijst naar een bepaalde *Exchange*.

*Columns*

* *currency\_id:* Een unieke letter combinatie die wordt gebruikt om currencies te kunnen identificeren. Wordt ook vaak een ‘symbol’ genoemd.
  + Data type is *Varchar*
  + Lengte is 6
  + Voorbeeld: BTC
* *exchange\_id:* Een foreign key die wijst naar een uniek identificeerbare rij in de *Exchange* tabel
* *trading\_symbol:* Een letter combinatie die wordt gebruikt voor het handelen in de currency. Kan hetzelfde of anders zijn dan de *currency\_id*.
  + Data type is *Varchar*
  + Lengte is 6
  + Voorbeeld: USDT
* *name:* De naam van de currency.
  + Data type is *Varchar*
  + Lengte is 25
  + Voorbeeld: Dogecoin
* *price\_change:* Hoeveel (positief of negatief) de prijs in het afgelopen interval (uur) is veranderd.
  + Genoteerd in ‘quote’ currency (EUR, USD)
  + Data type is *Decimal*
  + Lengte is 20,10
  + Voorbeeld: -1.095544
* *percent\_change:* Hoeveel (positief of negatief) de prijs in het afgelopen interval (uur) is veranderd.
  + Genoteerd in percentages (decimalen)
  + Data type is *Decimal*
  + Lengte is 6,2
  + Voorbeeld: -3,28
* *previous\_close\_price:* De close prijs van het vorige interval (uur).
  + Genoteerd in ‘base’ currency (DOGE)
  + Data type is *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 5,43434
* *last\_price:* De meest actuele prijs van de currency.
  + Genoteerd in ‘base’ currency (DOGE)
  + Data type is *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 5,43434

**Exchange tabel**

De *Exchange* tabel bevat informatie over een bepaalde exchange die wordt gebruikt voor het ophalen van data en het handelen.

*Relaties*

* De *Exchange* tabel heeft een één op meer (1:N) relatie met de *Candlestick* tabel. Elke candlestick data **moet** van een bepaalde exchange komen, terwijl van een exchange meerdere candlesticks **kunnen** komen.
* De *Exchange* tabel heeft een één op meer (1:N) relatie met de *Currency* tabel. Elke currency **moet** zijn data van een bepaalde exchange krijgen, terwijl van een exchange meerdere currencies kunnen komen.
* De *Exchange* tabel heeft een één op meer (1:N) relatie met de *Trade* tabel. Elke trade **moet** op een bepaalde exchange worden gedaan, terwijl op een exchange meerdere trades **kunnen** worden gedaan.

*Sleutels*

De *Exchange* tabel bevat geen foreign keys.

*Columns*

* *exchange\_id:* Een unieke letter combinatie die wordt gebruikt om exchanges te kunnen identificeren. Wordt ook vaak een ‘symbol’ genoemd.
  + Datatype is *Varchar*
  + Lengte is 6
  + Voorbeeld: BIN
* *name:* De naam van de exchange.
  + Datatype is *Varchar*
  + Lengte is 25
  + Voorbeeld: Binance

**Wallet tabel**

De *Wallet* tabel bevat informatie over de ‘wallet’ van de gebruiker. Vooral hoeveel de gebruiker van een bepaalde currency op het moment heeft.

*Relaties*

* De *Wallet* tabel heeft een één op meer (1:N) relatie met de *Trade* tabel. Elke trade **moet** gedaan zijn met een bepaalde wallet, terwijl er meerdere trades **kunnen** zijn gedaan met een wallet.

*Sleutels*

De *Wallet* tabel bevat geen foreign keys.

*Columns*

* *wallet\_id:* Een unieke combinatie van letters en cijfers die wordt gebruikt om een wallet mee te kunnen identificeren.
  + Genoteerd als surrogate key
  + Data type is *Char*
  + Lengte is 36
  + Voorbeeld: 79c9e1ac-96f1-11e5-85a6-000c29f1f6c4
* *eur\_value:* De waarde van de wallet in Euro’s
  + Data type is *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 530,70
* *usd\_value:* De waarde van de wallet in Dollars
  + Data type is *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 530,70
* *doge\_value:* De waarde van de wallet in Dogecoins
  + Data type is *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 5,76560

**Trade tabel**

De *Trade* tabel bevat informatie over een handeling (trade) die is gedaan. Dit kan zijn dat er iets is gekocht of verkocht.

*Relaties*

* De *Trade* tabel heeft een één op meer (1:N) relatie met de *Wallet* tabel. Elke trade **moet** gedaan zijn met een bepaalde wallet, terwijl er meerdere trades **kunnen** zijn gedaan met een wallet.
* De *Trade* tabel heeft een één op meer (1:N) relatie met de *Exchange* tabel. Elke trade **moet** gedaan zijn met een bepaalde exchange, terwijl er meerdere trades **kunnen** zijn gedaan met een exchange.
* De *Trade* tabel heeft een één op meer (1:N) relatie met de *Currency* tabel. Elke trade **moet** gedaan zijn met een bepaalde currency, terwijl er meerdere trades **kunnen** zijn gedaan met een currency.

*Sleutels*

* Elke *Trade* rij **moet** een foreign key bevatten die wijst naar een bepaale *Exchange*.
* Elke *Trade* rij **moet** een foreign key bevatten die wijst naar een bepaale *Currency*.
* Elke *Trade* rij **moet** een foreign key bevatten die wijst naar een bepaale *Wallet*.

*Columns*

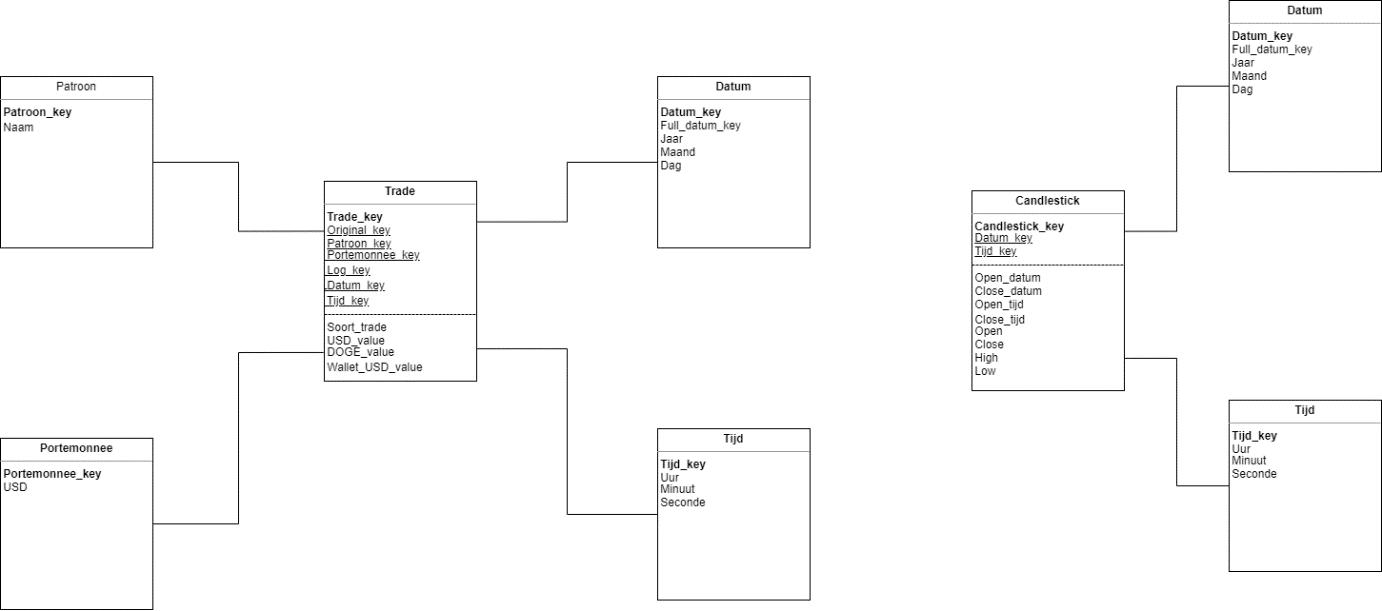
* *trade\_id:* Een unieke combinatie van letters en cijfers die wordt gebruikt om een trade mee te kunnen identificeren.
  + Genoteerd als surrogate key
  + Data type is *Char*
  + Lengte is 36
  + Voorbeeld: 79c9e1ac-96f1-11e5-85a6-000c29f1f6c4
* *exchange\_id:* Een foreign key die wijst naar een uniek identificeerbare rij in de *Exchange* tabel
* *currency\_id:* Een foreign key die wijst naar een uniek identificeerbare rij in de *Currency* tabel. Als er bijvoorbeeld voor €100,- aan Dogecoin is gekocht, dan verwijst deze foreign key naar de Dogecoin currency.
* *wallet\_id:* Een foreign key die wijst naar een uniek identificeerbare rij in de *Wallet* tabel
* *price:* De prijs van de currency op het moment van de trade.
  + Data type is een *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 5,06868
* *quantity:* De hoeveelheid die is gekocht of verkocht. Als je voor €100,- aan Dogecoin koopt, dan is dit de hoeveelheid die je krijgt.
  + Data type is *Integer*
  + Lengte is 9
  + Voorbeeld: 45
* *eur\_value:* De hoeveelheid dat de trade kostte of opleverde in Euro’s
  + Data type is *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 1050,00
* *usd\_value:* De hoeveelheid dat de trade kostte of opleverde in Dollars
  + Data type is *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 1050,00
* *doge\_value:* De hoeveelheid dat de trade kostte of opleverde in Dogecoin. Dit is het aantal dat is gekocht/verkocht maal de prijs van de currency op het moment van de trade.
  + Data type is *Decimal*
  + Lengte is 20,10
  + Voorbeeld: 6354,43534
* *status:* De huidige status van de trade. Dit kan bijvoorbeeld zijn dat het pending, failed, processed of completed is.
  + Genoteerd als één van een aantal statusberichten
  + Data type is *Varchar*
  + Lengte is 25
  + Voorbeeld: Pending
* *success:* Of de trade succesvol was of niet.
  + Genoteerd als getal (0 of 1)
  + Data type is *Integer*
  + Lengte is 1
  + Voorbeeld: 1
* *error\_code:* Een bepaalde code die zegt of er iets is misgegaan.
  + Genoteerd als http status code getallen
  + Data type is *Integer*
  + Lengte is 3
  + Voorbeeld: 200
* *error\_message:* Een bepaald bericht die zegt wat er is misgegaan.
  + Data type is *Varchar*
  + Lengte is 255
  + Voorbeeld: Not enough currency in wallet to make that trade
* *timestamp:* Het moment van de trade
  + Genoteerd als Epoch timestamp in millisecondes
  + Data type is *Integer*
  + Lengte is 13
  + Voorbeeld: 1234567890000
* *taker\_side:* Welke kant is gekozen (buyer of seller)
  + Genoteerd als getal (0 of 1)
  + Data type is *Integer*
  + Lengte is 1
  + Voorbeeld: 1

**Algemene kolommen**

Naast de bovenstaande kolommen bevat elke tabel ook de volgende twee kolommen:

* *last\_added:* Wanneer een rij is toegevoegd aan de tabel
  + Genoteerd als Epoch timestamp in millisecondes
  + Data type is *Integer*
  + Lengte is 13
  + Voorbeeld: 1234567890000
* *last\_updated:* Wanneer een rij is gewijzigd in de tabel
  + Genoteerd als Epoch timestamp in millisecondes
  + Data type is *Integer*
  + Lengte is 13
  + Voorbeeld: 1234567890000

**Data warehouse TDM**



**Toelichting**

In het bovenstaande figuur zie je de datamarts waar wij analyses op baseren. In het midden staat de feit ook wel measure genoemd met daarom heen de dimensies die de feit ondersteunen. Doormiddel van de foreignkey relatie kunnen we ze aanroepen.

# Ontwerpbeslissingen

**Samenhang componenten**

Onze BI solution bestaat in feite uit drie basis componenten: De data pipeline, een Flask API en een VueJS frontend. Deze keuze is gemaakt, omdat op deze manier er een duidelijke splitsing qua taken en verantwoordelijkheden ontstaan; De frontend hoeft zich geen zorgen te maken over de connectie met de database, de Flask API hoeft zich niet bezig te houden met het weergeven van de data, etc. Dit creëert een goed scalable en onderhoudbare architectuur.

Er is gekozen voor Flask voor de API, omdat Flask erg fijn is om mee te werken wanneer er gebruik wordt gemaakt van redelijk kleine datasets en een MySQL database omgeving. De library’s die hiervoor bestaan zijn uitgebreid en goed onderhouden, wat Flask een goede keuze maakt.

Qua frontend is de keuze gevallen op VueJS, omdat uit het vorige project bleek dat de frontend niet te doen was in Flask. Je was dan ontzettend gelimiteerd, wat het voldoen aan de eisen erg lastig zou maken. Met VueJS is het een stuk gemakkelijker om een uitgebreid dashboard te maken, beter dan welk ander framework dan ook, mede vanwege verscheidene CSS frameworks die goed met VueJS werken, zoals Vuetify.

**Ontwerp code**

Er zijn een stel afspraken gemaakt over het ontwerp en ontwikkelen van de code. Ten eerste wordt bij de data pipeline dezelfde mappenstructuur als die van afgelopen project gevolgd. Hierbij hoeven deze keer geen classes gebruikt te worden. De code en de bestandsnamen moeten volledig in het Engels en de basis ontwerpprincipes, zoals SRP, DRY, etc. moeten zoveel mogelijk worden aangehouden. Functies mogen niet langer zijn dan 20 regels LOC om zo de code overzichtelijk te houden. Bij alle onduidelijkheden moet commentaar staan om onduidelijkheden uit de wereld te helpen.

**Keuze data warehouse architectuur**

Zoals beschreven in het globale ontwerp zijn er meerdere voordelen die ertoe hebben geleidt dat wij hebben gekozen voor Kimball’s data marts als architectuur. Om binnen een magere 8 weken een goed product neer te kunnen zetten en de complexiteit in gedachte te houden zijn architecturen als EDW’s, data lakes en data vaults geen optie. Kimball’s data marts met conformed dimensions bieden een simpele en snelle oplossing aan die voldoen aan al onze eisen en de requirements van de opdrachtgever.