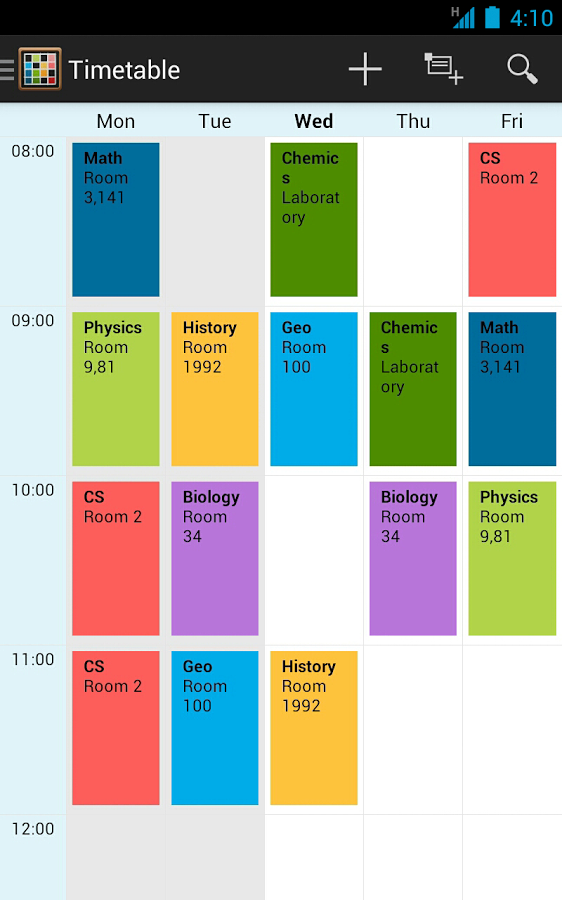
Architectuur document

Planning Maker



Project: Planning Maker

Auteur: Lesley Peters

Opdrachtgever: Frank Peters

Versie: 1.0

Versiedatum: 21-5-2017

**Documenthistorie**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Versie | Wijzigingen | Auteur | Datum | Verzendlijst | Verzoek |
| 0.1 | Opzet document | Nico Kuijpers | 22 apr 2014 |  |  |
| 0.2 | Opdeling componenten verplaatst naar tweede hoofdstuk; hoofdstukken communicatie en persistentie verwisseld; hoofdstuk detailed design toegevoegd | Nico Kuijpers | 21 sep 2016 |  |  |
| 0.3 | Inleiding toegevoegd, indeling verder uitgewerkt | Pieter Beukelman | mei 2017 |  |  |
| 0.4 | Domeinmodel & deployment diagram toegevoegd | Lesley Peters | 8 mei 2017 |  |  |

**Inhoud**

[**Documenthistorie**](#_30j0zll) **2**

[**1 Inleiding**](#_1fob9te) **4**

[1.1 Context](#_3znysh7) 4

[1.2 Applicatie](#_2et92p0) 4

[1.3 Niet-functionele eisen](#_tyjcwt) 4

[1.4 Doel van dit document](#_3dy6vkm) 4

[**2 Domeinmodel**](#_1t3h5sf) **5**

[2.1 Klassendiagram](#_4d34og8) 5

[2.2 Afbakening](#_2s8eyo1) 5

[**3 Opdeling in Componenten**](#_17dp8vu) **6**

[3.1 Componenten diagram](#_3rdcrjn) 7

[3.2 Koppeling tussen componenten](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.32hioqz) [7](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.32hioqz)

[3.3 Synchronisatie tussen componenten](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.1hmsyys) [7](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.1hmsyys)

[3.4 Services per component](#_35nkun2) 7

[3.5 Allocatie van objecten](#_1ksv4uv) [7](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.41mghml)

[3.6 Package structuur](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.2grqrue) [7](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.2grqrue)

[**4 Communicatie tussen componenten**](#_z337ya) **8**

[4.1 Klassendiagram tbv communicatie](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.vx1227) [8](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.vx1227)

[**5 Persistentie per component**](#_1y810tw) **9**

[**6 Detailed design per component**](#_2bn6wsx) **10**

[**7 Realisatie niet-functionele eisen**](#_qsh70q) **11**

[7.1 Betrouwbaarheid](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.3as4poj) [11](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.3as4poj)

[7.2 Performance](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.1pxezwc) [11](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.1pxezwc)

[7.3 Beveiliging](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.2p2csry) [11](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.2p2csry)

[7.4 Schaalbaarheid](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.147n2zr) [11](https://docs.google.com/document/d/11Uha1modT0CwDr7goGaZxgI9qAfcnVVHamZFYlL9AX4/edit#heading=h.147n2zr)

[**8 Deployment**](#_3o7alnk) **12**

[8.1 Deployment diagram](#_23ckvvd) 12

[**9 Specificatie van interfaces**](#_ihv636) **13**

# **1 Inleiding**

#### **1.1 Context**

Dit architectuurdocument heeft betrekking op de applicatie ‘Planning Maker’. Deze applicatie wordt gemaakt door Lesley Peters op de Fontys hogeschool. Deze applicatie word gemaakt voor de persoonlijke opdracht voor het vak GSO, hiervoor moet een gedistribueerde applicatie gemaakt worden met tenminste 3 verschillende componenten om onze leerdoelen aan te tonen.

#### **1.2 Applicatie**

Planning Maker is een applicatie waarbij gebruikers hun eigen agenda bij kunnen houden door taken en events toe te voegen. Daarnaast is het mogelijk om andere accounts uit te nodigen voor een gedeelde agenda, waarbij de uitgenodigde accounts taken en events kunnen toevoegen. De gedeelde agenda’s worden uiteindelijk samengevoegd waarna ieder account gemakkelijk zijn eigen taken en events kan terug zien. Veranderingen in een gedeelde agenda worden real-time gewijzigd voor andere gebruikers.

#### **1.3 Niet-functionele eisen**

De software moet Multi platform zijn. Wijzigingen aan een agenda moeten real-time zichtbaar zijn voor alle accounts die deelnemen aan deze agenda. Ook moet de database een uptime hebben van minstens 99%. Agenda’s mogen alleen ingezien worden als het een prive agenda is van de ingelogde gebruiker, of als een gebruiker uitgenodigd is voor de betreffende agenda.

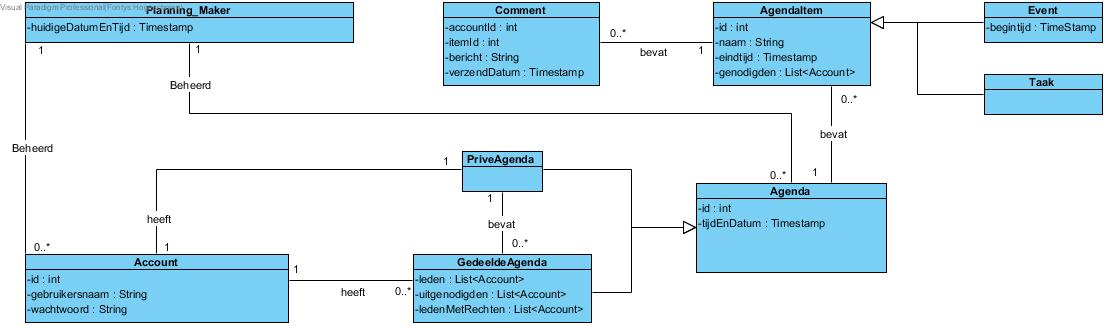
#### **1.4 Doel van dit document**

Dit document is opgezet om duidelijk te maken hoe de applicatie is opgebouwd. Dit word gedaan door gebruik te maken van klassendiagrammen, sequence diagrammen, een componenten diagram en een deployment diagram. In dit document word alleen het ontwerp besproken, voor meer informatie over de applicatie of de requirements word u verwezen naar het URS.

# **2 Domeinmodel**

*In dit hoofdstuk worden een of meer klassendiagrammen van het domein getoond. Daarnaast wordt een toelichting gegeven en worden gemaakte keuzes verantwoord. Tot slot wordt de afbakening van het project besproken.*

#### **2.1 Klassendiagram**

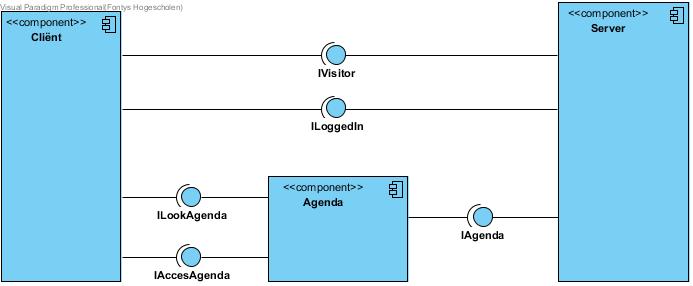
Figuur 1: Klassendiagram van het domein.

#### **2.2 Afbakening**

Aangezien de applicatie nog in ontwikkeling is kunnen sommige klassen en waardes nog veranderen, indien dit gebeurt zal het domein model aangepast worden en daarna ook dit document geupdate worden. De operaties zijn niet weergegeven om de overzichtelijkheid van het klassendiagram te bewaren.

# **3 Opdeling in Componenten**

#### **3.1 Componenten diagram**



Figuur 1: Componenten diagram.

In het componenten diagram staat op een simpele manier beschreven hoe de communicatie tussen de componenten te werk gaat. Zoals je in figuur 1 kan zien zijn dat 3 componenten.

#### **3.2 Koppeling tussen componenten**

Voor dit project is er gekozen om de communicatie te leggen via RMI. Hierbij zijn er 3 componenten die via de server met elkaar in verbinding staan. De server zal voornamelijk informatie opslaan en uitdelen, de client vraagt informatie op van de server, en de client die als eerste een groeps agenda opent, die host ook het agenda component dat dan beschikbaar wordt voor andere clients. Dit kan worden gedaan door gebruik te maken de interfaces die te zien zijn in Figuur 1.

#### **3.3 Synchronisatie tussen componenten**

Ten behoeve van de snelheid van de applicatie is het van belang om de juiste informatie door te geven. Door zo min mogelijk informatie door te geven wordt de tijdsynchronisatie tussen de componenten verbeterd.

#### 

#### **3.4 Services per component**

In de onderstaande tekst staat beschreven wat de functies zijn van elke interface. Dit zijn de interfaces die bij de koppeling tussen de componenten horen.

**IVisitor**

* Functie om in te loggen
* Functie om een account aan te maken

**ILoggedIn**

* Functie om uit te loggen
* Functie om een privé agenda in te zien
* Functie om een event toe te voegen aan een privé agenda
* Functie om een taak toe te voegen aan een privé agenda
* Functie om commentaar toe te voegen aan een event of taak
* Functie om een gedeelde agenda aan te maken

**IAgenda**

* Functie om een gedeelde agenda op te halen van de server
* Functie om een gedeelde agenda op te slaan op de server

**ILookAgenda**

* Functie om een gedeelde agenda in te zien
* Functie om een comment te plaatsen aan taak of event
* Functie om een comment te verwijderen van een taak of event
* Functie om een gedeelde agenda te verlaten

**IAccesAgenda**

* Functie om een account toe te voegen aan een gedeelde agenda
* Functie om een account te verwijderen uit een gedeelde agenda
* Functie om een event of taak toe te voegen aan een gedeelde agenda
* Functie om een gebruiker te koppelen aan een taak of event
* Functie om een gebruiker te ontkoppelen van een taak of event

#### 

#### 

#### **3.5 Allocatie van objecten**

IVisitor : Krijgt de functies om in te loggen of te registreren aangeboden van de server, deze interface word ontvangen tijdens het opstarten van de applicatie

ILoggedIn: Maakt het mogelijk om persoonlijke acties uit te voeren zoals taken toevoegen/verwijderen van een persoonlijke agenda of een gedeelde agenda aan te maken. Deze interface word ontvangen zodra een gebruiker ingelogd is.  
IAgenda : Heeft de mogelijkheid om agenda’s in te laden of op te slaan naar de server. Deze interface word ontvangen zodra een gebruiker inlogd of een gedeelde agenda opent.

ILookAgenda: Geeft de mogelijkheid om een gedeelda agenda in te zien en om commentaar onder taken/events toe te voegen. Deze interface word verkregen zodra een gedeelde agenda wordt geopend.

IAccesAgenda: Geeft de mogelijkheid om een gedeelde agenda te bewerken. Deze interface wordt verkregen als een gedeelde agenda word geopend en de gebruiker rechten heeft om de agenda aan te passen.

#### **3.6 Package structuur**

De applicatie is opgedeeld in vier packages: client, server, agenda en shared. In de client package worden alle klassen geplaatst waar de client gebruik van maakt. In de server package worden alle klassen geplaatst waar de server gebruikt van maakt. In de agenda package worden alle klassen geplaatst waar het agenda component gebruik van maakt en in de shared package worden alle klassen geplaatst die gedeeld worden door de verschillende componenten.gel

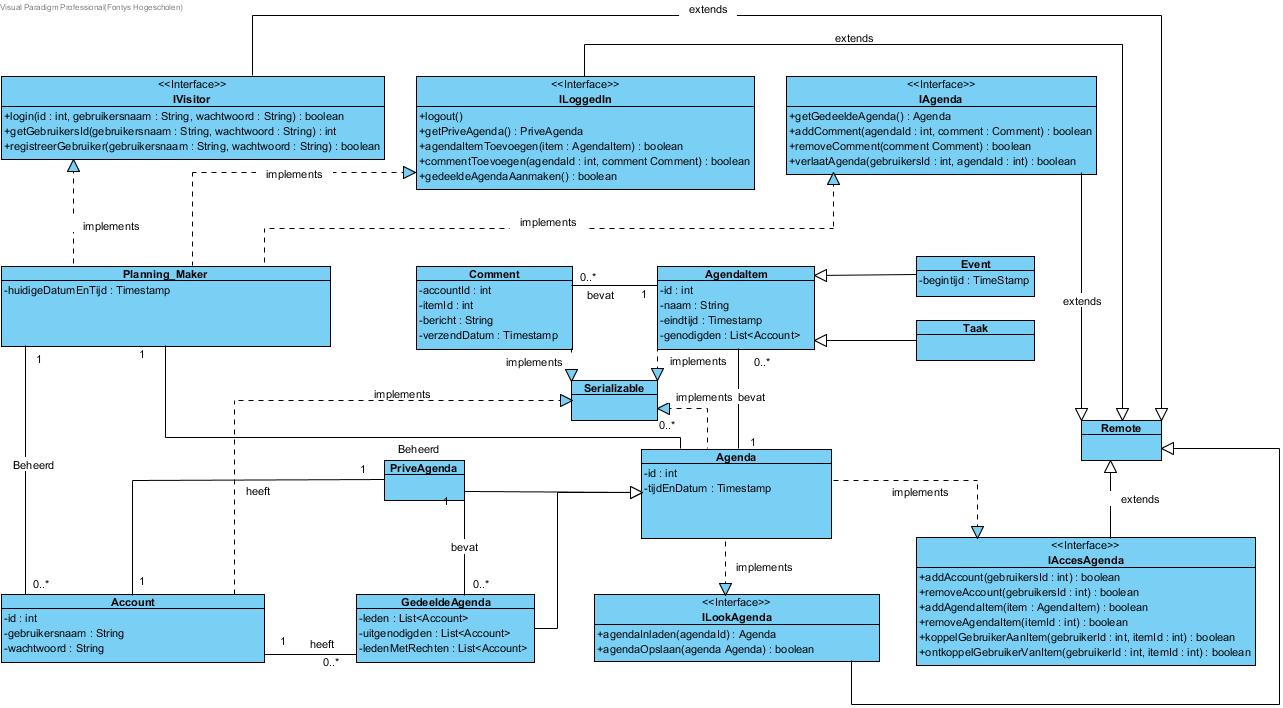
# 

# 

# **4 Communicatie tussen componenten**

## **4.1 Klassendiagram t.b.v. communicatie**

Hieronder wordt het klassendiagram weergegeven met de interfaces toegevoegd.



Figuur 1: [Klassendiagram](https://gyazo.com/0fa4c84e19027415aeb1c5a27db7e506).

Hierboven zie je het klassendiagram met de interfaces toegepast. Daarnaast is aangegeven welke interfaces remote zijn. Ook worden de klassen die deze interfaces implementeren unicastRemoteObject, dit is echter niet in het klassendiagram geplaatst om het enigszins overzichtelijk te houden.

# **5 Persistentie per component**

Om bestanden veilig op te slaan, ook als er een storing mocht zijn in de server, word er gebruik gemaakt van een database zodat alle gegevens opgeslagen blijven.

#### **5.1 Database**

De server is het enige component dat in verbinding staat met de database, de cliënt en agenda kunnen dus alleen via de server verbinding maken met de database. Hieronder wordt uitgelegd door middel van welke tabellen de gegevens worden opgeslagen in de database.

De cliënt heeft alleen een verbinding met de server alle informatie die hij nodig heeft krijgt hij van de server binnen met gebruik van RMI, de server daarentegen heeft verbinding met de Database. De server houd alles bij op zichzelf en update de database waar nodig. De database is nodig om, bij uitval van de server. de server weer zonder problemen kunnen op te starten. ook kan de database gebruikt worden op het moment dat er meerdere servers worden aangemeld.

Gebruiker: id, gebruikersnaam, wachtwoord, email

Deze tabel bevat de gebruikers informatie

Agenda: id, isgedeeld

Deze tabel bevat de agenda’s en een boolean om te checken of een agenda privé

of gedeeld is

AgendaItem: id, naam, omschrijving, begintijd, eindtijd

Deze tabel bevat alle taken en events. indien begintijd ’null’ is dan gaat het om een taak, anders om een event.

Gebruiker\_Agendas: gebruikerId, AgendaId

Deze tabel bevat gebruikerId’s die gekoppeld zijn aan een AgendaId

Agenda\_AgendaItems: agendaItemId, agendaId

Deze tabel bevat de agendaId’s gekoppeld zijn aan agenda items

Comment: gebruikerid, agendaItemId, comment, Time

Deze tabel bevat comments die door gebruikers zijn geschreven en gekoppeld zijn aan een agenda item

#### 

#### 

# **6 Detailed design per component**

AgendaComponent

Via het agenda component bestaat er een communicatie tussen een gedeelde agenda, clients en de server. Dit component kan door een cliënt gestart worden zodra hij een gedeelde agenda opent, alle gebruikers die daarna dezelfde agenda openen zullen in verbinding staan met dit agenda component. Via het agenda component worden alle wijzigingen van clients aan de agenda afgehandeld, en worden daarna verstuurd naar de server.

ServerComponent

Via het server component worden alle gebruikers, agenda’s, taken, events en comments bijgehouden.

Het server component is het enige component dat een verbinding heeft met de database. Tijdens het opstarten van dit component worden alle gegevens van de database ingelezen zodat deze gegevens snel uit te delen zijn aan andere componenten.

ClientComponent

Dit component is beschikbaar voor alle gebruikers en zorgt voor de afhandeling van de user inputs. Alle acties die een gebruiker maakt zullen doorgestuurd worden naar de server en worden daar afgehandeld.

Daarnaast zal de gebruiker via het Client component een Agenda component kunnen opstarten waardoor er 2 componenten tegelijkertijd kunnen draaien op 1 systeem.

# 

# 

# **7 Realisatie niet-functionele eisen**

Hieronder zullen de niet-functionele eisen worden besproken.

**7.1 Betrouwbaarheid**

- De database moet een uptime van 99% hebben.

- Indien de database down gaat, moet er een back-up server zijn, en hetzelfde

geldt voor de applicatie indien een gebruiker uitvalt.

Omdat alle gegevens die voor een gebruiker van belang zijn, zijn opgeslagen in een database is het van groot belang dat de database vrijwel altijd toegankelijk moet zijn. Dit is te bereiken door een tweede back-up database aan te maken, die het werk overneemt zodra de eerste database niet beschikbaar is.

#### **7.2 Performance**

- De applicatie moet realtime zijn.

Aangezien er meerdere gebruikers tegelijkertijd een agenda kunnen inzien en bewerken is het noodzakelijk dat de gebruikers wijzigingen real-time kunnen zien. Dit word mogelijk gemaakt om één gebruiker die als eerste een gedeelde agenda opent het Agenda Component (zie hoofdstuk 6 AgendaComponent) te laten hosten. Gebruikers die daarna dezelfde agenda willen bekijken worden dan direct met de host in verbinding gebracht waardoor er een real-time verbinding ontstaat tussen de clients die dezelfde agenda bezichtigen.

#### **7.3 Beveiliging**

- Een privé agenda mag alleen ingezien worden door de eigenaar

Om te garanderen dat niemand de agenda kan inzien van iemand anders worden agenda’s in de database gekoppeld aan een gebruikers ID. Hierdoor kan alleen een ingelogde gebruiker zijn eigen agenda ophalen uit de database. Daarnaast wordt voor het inzien van een privé agenda ook niet gebruik gemaakt van het apparte agenda component (zie hoofdstuk 6 AgendaComponent).

- Een gedeelde agenda mag alleen ingezien worden door uitgenodigde gebruikers

Om dit te garanderen is er ook hier gebruik gemaakt van een aparte koppeltabel in de database, die gebruikers koppelt aan een agenda. Indien het ID van een ingelogde gebruiker en het ID van een agenda niet voorkomen in deze tabel dan krijgt de gebruiker geen toegang tot de gedeelde agenda.

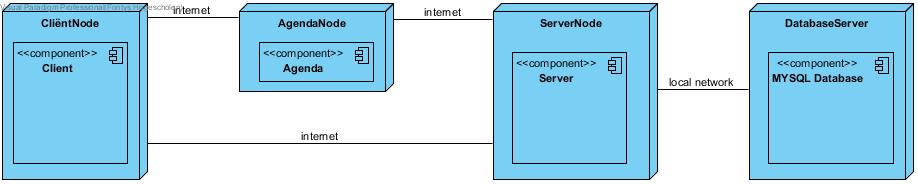
#### **7.4 Schaalbaarheid**

- het programma moet Multi platform zijn.

Aangezien de applicatie in JAVA word gemaakt is er automatisch de mogelijkheid om de applicatie vanuit meerdere systemen te openen.

# **8 Deployment**

#### **8.1 Deployment diagram**



Figuur 1: Deployment diagram.  
  
De Client node zal beschikbaar worden voor alle gebruikers van de applicatie, deze bevat alleen de client component met de user interface en de mogelijkheid om te communiceren met de server of het agenda component.

De Server node zal alleen beschikbaar zijn voor de eigenaar en ontwikkelaars van deze applicatie. Deze node bevat de server component die informatie van de clients verwerkt en opslaat in de database.

De Agenda node zal door een client worden aangemaakt zodra een gedeelde agenda word geopent. Zodra meerdere clients dezelfde gedeelde agenda willen inzien dan worden deze verbonden met dezelfde agenda component dat de eerste gebruiker aangemaakt heeft. Verder zorgd deze node voor de communicatie met de server om een agenda in te laden of om wijzigingen op te slaan.

# **9 Specificatie van interfaces**

*In dit hoofdstuk wordt de specificatie van interfaces besproken. Voor ieder interface wordt per methode gedefinieerd:*

*IVisitor*

* *getGebruikersId methode*
  + *getGebruikersId(String gebruikersnaam, String wachtwoord) : int*
  + *returned een int dat het gebruikers ID bevat indien de gebruiker bestaat in de database*
  + *Deze methode zal controleren of de gebruikersnaam voorkomt in de database en of het meegegeven wachtwoord correct is. Indien de gegevens correct zijn zal deze methode het bijbehorende gebruikers ID teruggeven, indien de gegevens niet kloppen wordt 0 gereturned.*
  + *Er kan een EmptyFieldException opgeroepen worden als de gebruikersnaam of wachtwoord niet ingevuld zijn geworden.*
* *login methode*
  + *login(int id, String gebruikersnaam, String wachtwoord) : ILoggedIn*
  + *returned een boolean; true indien het inloggen succesvol is, false indien het inloggen niet gelukt is*
  + *Deze methode zal de gebruiker (Account) initialiseren en doorsturen naar het hoofdmenu, er hoeven verder geen checks meer worden uitgevoerd aangezien dit al gebeurd is in de getGebruikersId methode.*
* *registreer methode*
  + *registreerGebruiker(String gebruikersnaam, String wachtwoord) : boolean*
  + *returned een boolean; true indien de registratie gelukt is, false indien de registratie niet gelukt is.*
  + *Deze methode zal proberen een account te registreren in de database met het meegegeven gebruikersnaam en wachtwoord. Indien de gebruikersnaam al bestaat zal dit geweigerd worden.*
  + *Deze methode kan een EmptyFieldException oproepen indien de gebruikersnaam of wachtwoord niet ingevuld zijn.*

*ILoggedIn*

* *logout methode*
  + *logout() : void*
  + *Deze methode zal de sessie met de gebruiker verbreken en de gebruiker terugsturen naar het inlog scherm*
* *getPriveAgenda methode*
  + *getPriveAgenda : PriveAgenda*
  + *returned de prive agenda van een gebruiker inclusief alle taken en events van gedeelde agenda’s*
  + *Deze methode zal de prive agenda van de gebruiker ophalen en daarnaast alle taken en events van gedeelde agenda’s toevoegen. Deze agenda word dan gereturned.*
* *agendaItemToevoegen methode*
  + *agendaItemToevoegen(AgendaItem item, int gebruikersId, int agendaId) : boolean*
  + *returned een boolean; true indien het toevoegen van een agenda item gelukt is, false indien het toevoegen mislukt is.*
  + *Deze methode zal een agenda item proberen toe te voegen in de database voor een prive agenda.*
* *commentToevoegen methode*
  + *commentToevoegen(int agendaItemId, int gebruikersId, Comment comment) : boolean*
  + *returned true indien het toevoegen gelukt is, returned false indien het toevoegen van de comment niet gelukt is.*
  + *Deze methode zal een comment toevoegen aan een agenda item.*
* *gedeeldeAgendaAanmaken methode*
  + *gedeeldeAgendaAanmaken(String naam) : boolean*
  + *returned true indien de agenda aangemaakt is geworden, false indien dit niet gelukt is.*
  + *Deze methode zal een nieuwe gedeelde agenda aanmaken en toevoegen in de database, daarna wordt de gebruiker doorgestuurd naar deze nieuwe agenda.*

*IAgenda*

* *getGedeeldeAgenda methode*
  + *getGedeeldeAgenda(int agendaId) : Agenda*
  + *returned een gedeelde agenda*
  + *Deze methode zal controleren of iemand al een agenda geopend heeft met hetzelfde id, als dit niet het geval is zal er een nieuw component gestart worden om deze agenda te hosten. Als wel al iemand deze agenda open heeft staan dan wordt een verbinding gemaakt naar de bestaande agenda.*
* *addComment methode*
  + *addComment(int gebruikersId, int agendaItemId,Comment comment) : boolean*
  + *returned true indien het toevoegen van de comment gelukt is, false indien dit niet gelukt is*
  + *Deze methode zal een comment toevoegen van een gebruiker aan een agenda item voor een gedeelde agenda*
* *removeComment methode*
  + *removeComment(int commentId) : boolean*
  + *returned true indien de comment verwijderd is, false indien dit niet gelukt is*
  + *Deze methode zal een comment verwijderen met een gegeven comment id, ook word er van te voren gecheckt of de comment is geplaatst door de ingelogde gebruiker.*
* *verlaatAgenda methode*
  + *verlaatAgenda(int gebruikersId, int agendaId) : boolean*
  + *returned true indien het verlaten gelukt is, false indien dit niet gelukt is*
  + *Deze methode zal de ingelogde gebruiker verwijderen van de gedeelde agenda die meegegeven word.*

*ILookAgenda*

* *agendaInladen methode*
  + *agendaInladen(int agendaId) : Agenda*
  + *returned een gedeelde agenda met het meegegeven agenda id*
  + *Deze methode zal een gedeelde agenda ophalen uit de database en returnen zodat de gebruiker hem kan inzien*
* *agendaOpslaan methode*
  + *agendaOpslaan(Agenda agenda) : boolean*
  + *returned true indien het opslaan gelukt is geworden of false indien het opslaan mislukt is*
  + *Deze methode zal een agenda naar de server sturen waar het verwerkt en opgeslagen word in de database*

*IAccesAgenda*

* *addAccount methode*
  + *addAccount(int gebruikersId) : boolean*
  + *returned true indien het toevoegen gelukt is, false indien dit niet gelukt is*
  + *Deze methode zal een gebruiker met het gegeven gebruikersId toevoegen aan een gedeelde agenda*
* *removeAccount methode*
  + *removeAccount(int gebruikersId) : boolean*
  + *returned true indien het verwijderen gelukt is, false indien dit niet gelukt is*
  + *Deze methode zal een gebruiker uit een gedeelde agenda verwijderen*
* *addAgendaItem methode*
  + *addAgendaItem(int agendaId, AgendaItem item) : boolean*
  + *returned true indien het toevoegen is gelukt, false indien dit niet is gelukt*
  + *Deze methode zal een agenda item toevoegen aan een gedeelde agenda van het meegegeven agenda id*
* *removeAgendaItem methode*
  + *removeAgendaItem(int agendaItemId) : boolean*
  + *returned true indien het verwijderen is gelukt, false indien dit niet is gelukt*
  + *Deze methode zal een agenda item met het meegegeven agendaItem id verwijderen uit de database*
* *koppelGebruikerAanItem methode*
  + *koppelGebruikerAanItem(int gebruikerId, int itemId) : boolean*
  + *returned true indien het koppelen gelukt is, false indien dit niet gelukt is*
  + *Deze methode zal een gebruiker koppelen aan een agenda item*
* *ontkoppelGebruikerVanItem methode*
  + *ontkoppelGebruikerVanItem(int gebruikerId, int itemId) : boolean*
  + *returned true indien het ontkoppelen gelukt is, false indien dit niet gelukt is*
  + *Deze methode zal een gebruiker ontkoppelen van een agenda item*