1-11-2018

Klaas van der Linden en Luca Hogeweide

HAN OOSE-A

Software Design Description Odinido

Case study OOAD OOSE

1. Inhoud

[2 Introduction 2](#_Toc528784444)

[2.1 Opdracht omschrijving 2](#_Toc528784445)

[2.2 Doel document 2](#_Toc528784446)

[2.3 Definitions, acronyms, and abbreviations 2](#_Toc528784447)

[3 Detailed Design Description 3](#_Toc528784448)

[Odinido 3](#_Toc528784449)

[3.1 Design Class Diagram 3](#_Toc528784450)

[3.2 Sequence Diagrams 3](#_Toc528784451)

[Starten Kennistoets 3](#_Toc528784452)

[Genereren score 4](#_Toc528784453)

[Aanmaken kennistoets 4](#_Toc528784454)

[Registeren 5](#_Toc528784455)

[Uitvoeren Kennistoets 5](#_Toc528784456)

[Uitvoeren Teamgevecht 6](#_Toc528784457)

[3.2.1 Activity and State Diagrams 6](#_Toc528784458)

[3.2.2 Design decisions made for the sub-system 6](#_Toc528784459)

[Bijlage 7](#_Toc528784460)

# Introduction

## Opdracht omschrijving

Voor een product omschrijving zie [SRS 2.1.](SRS%20Odinido%20Luca%20Hogeweide%20en%20Klaas%20van%20der%20Linden.docx)

## Doel document

In dit document staan de design en analyse onderdelen die relevant zijn voor de software. Hiermee kan de programmeur zien wat er gedaan moet worden om de applicatie te realiseren.

## Definitions, acronyms, and abbreviations

|  |  |
| --- | --- |
| Term | Description |
|  | Afkortingen die in document optreden hier benoemen |

# Detailed Design Description

## Odinido

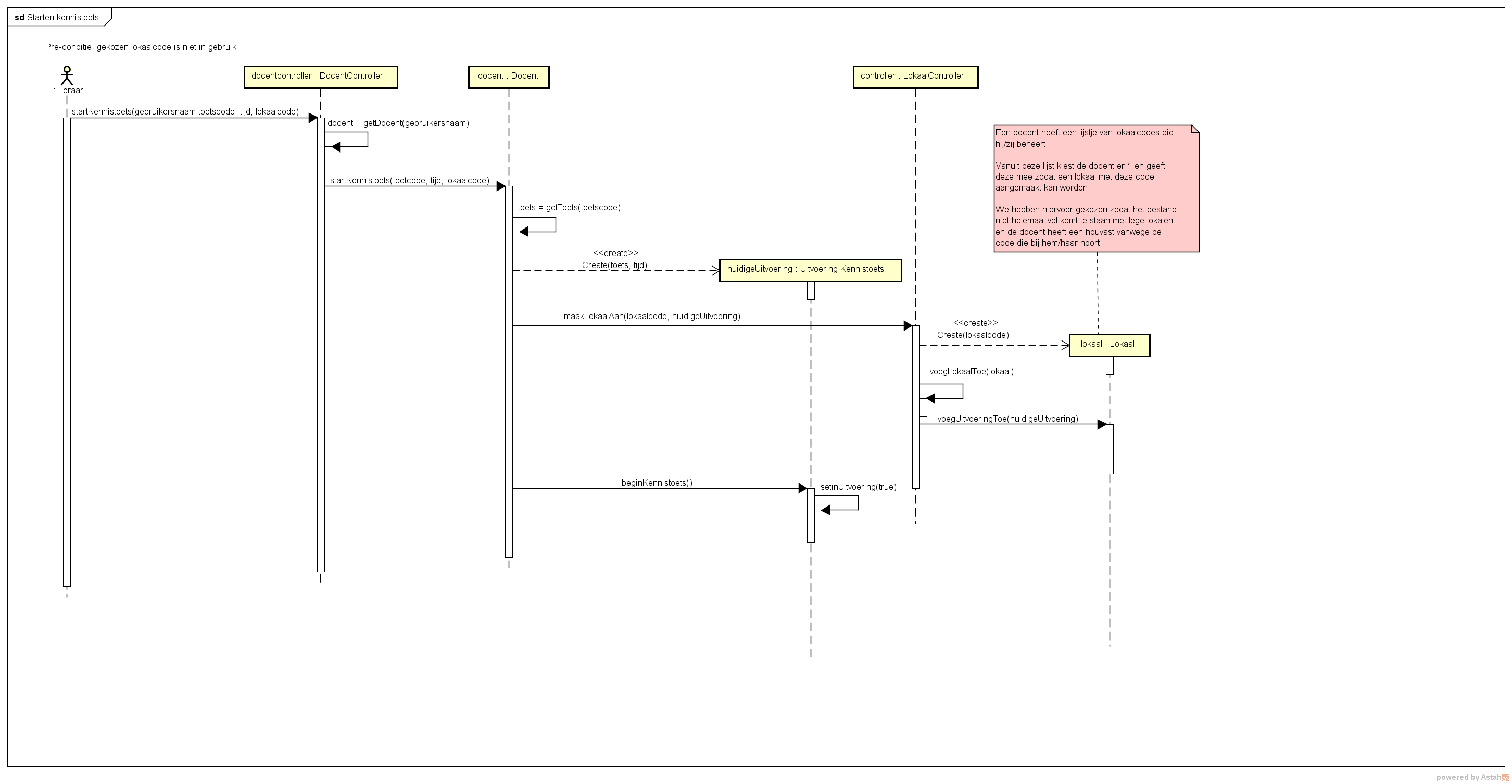
## Design Class Diagram

Zie bijlage voor de afbeelding van het Design Class Diagram.

## Sequence Diagrams

Hieronder bevinden zich alle sequence diagrammen inclusief eventuele toelichting en argumentatie voor de gemaakte keuzes. Op alle sequence diagrammen kan het control + click commando worden uitgevoerd om zo een JPEG variant van het diagram te openen waarop kan worden ingezoomd om het diagram beter te bekijken.

### 3.2.1 Starten Kennistoets

[](Sequence%20Diagrammen/Starten%20kennistoets.png)

Figuur 1 Sequence diagram starten kennistoets

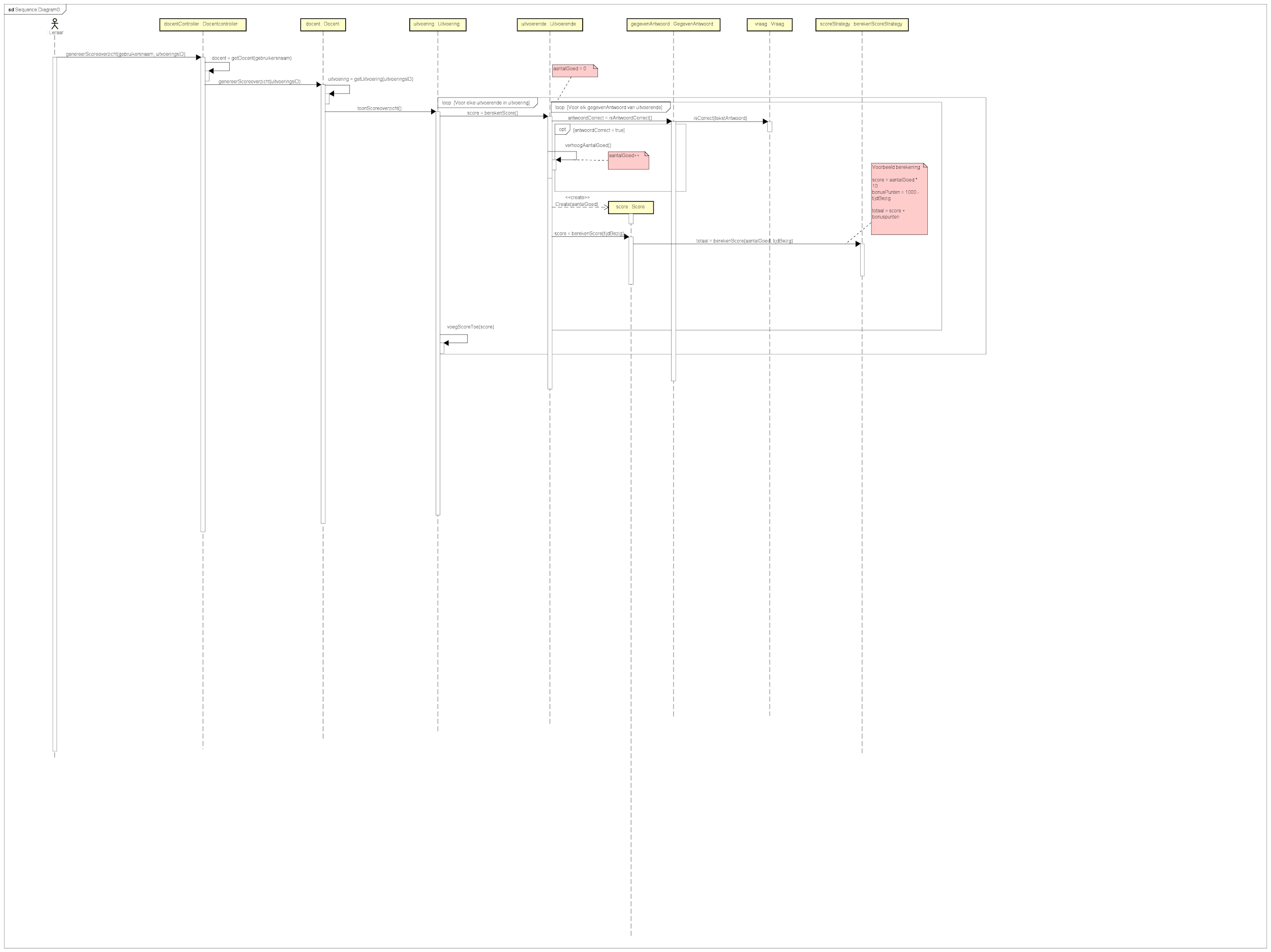
De system call van startenkennistoets komt binnen bij de docentController. Deze controller beheert alle docenten en vraagt op basis van de meegegeven gebruikersnaam de juiste docent op. De controller wordt alleen gebruikt voor binnen komende calls van de actor en voor het ophalen van het juiste object. De verdere taken worden door gedelegeerd naar het opgehaalde object.

De docent haalt bij zichzelf de kennistoets op, op basis van de meegegeven toetscode. Daarna maakt de docent een nieuwe uitvoering aan met de juiste paramaters. Hierbij wordt het creator pattern toegepast. De docent beschikt over de juiste initialisatie data om de uitvoering aan te maken en beheert de uitvoering, dus is deze klasse het meest geschikt om de uitvoering aan te maken.

Ook moet er een lokaal worden aangemaakt waarin de uitvoering zich afspeelt. Deze taak delegeert de docent door naar de lokaalcontroller. Deze beheert namelijk de lokalen. Ook hier word dust het creator pattern toegepast. De controller voegt het lokaal aan zichzelf toe en de meegegeven uitvoering aan het lokaal. Als dit gedaan is begint de docent de uitvoering.

### 

### 3.2.2 Genereren score

[](Sequence%20Diagrammen/Genereren%20score.jpg)

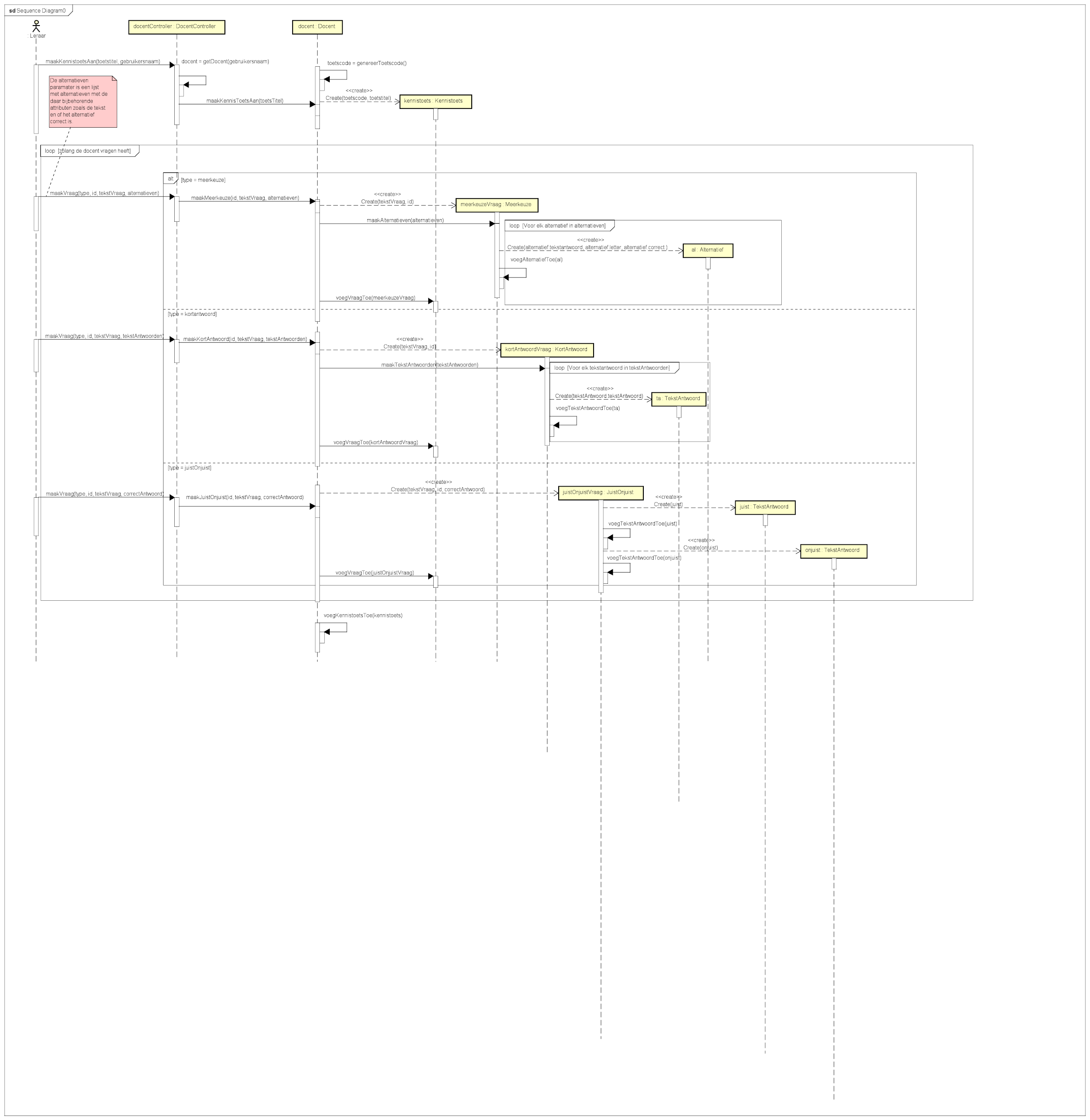
Figuur 2 Sequence diagram score genereren

Bij het genereren van de score wordt het pattern information expert toegepast. Elke klasse krijgt voor het berekenen van de score de juiste verantwoordelijkheid toegewezen op basis van welke data de klasse bevat. Dit zorgt ervoor dat information hiding in stand wordt gehouden. Klassen weten namelijk niet van elkaar hoe ze de score berekenen. Bovendien houdt dit de koppeling tussen de klasse laag en de cohesie hoog.

De uitvoerende maakt de score. Deze bevat namelijk de gegevenAntwoorden. Aan de gegevenAntwoorden kan de taak worden door gedelegeerd om te kijken of het antwoord van de student overeenstemt met het beantwoorden van de vraag. Deze delegeert de taak door naar de bijbehorende vraag. De vraag bevat een method isCorrect waarbij de implementatie afhankelijk is van het type. Om dit op te lossen is er gebruik gemaakt van polymorphism waarbij vraag een abstracte klasse is en ieder type klasse van vraag een eigen implementatie heeft voor het kijken of het gegeven antwoord correct is.

Op basis hiervan kan de uitvoerende het aantal goede antwoorden ophogen en hiermee een nieuwe score object creëren. Het score object berekent vervolgens zelf de score met behulp van het strategy pattern. Het strategy pattern is hier toegepast zodat de strategie voor het berekenen van de score gemakkelijk kan worden aangepast en zodat er in de toekomst eenvoudig nieuwe score strategieën kunnen worden toegevoegd, aan gezien de punten systematiek bij Odinido op het moment nog niet definitief is.

### 3.2.3 Aanmaken kennistoets

[](Sequence%20Diagrammen/aanmakenKennistoets.png)

Figuur 3 Sequence diagram aanmaken kennistoets

Bij het maken van een kennistoets is ervoor gekozen om gebruik te maken van verschillende system calls van de actor om zo het maken van de kennistoets en de vragen te onderscheiden.

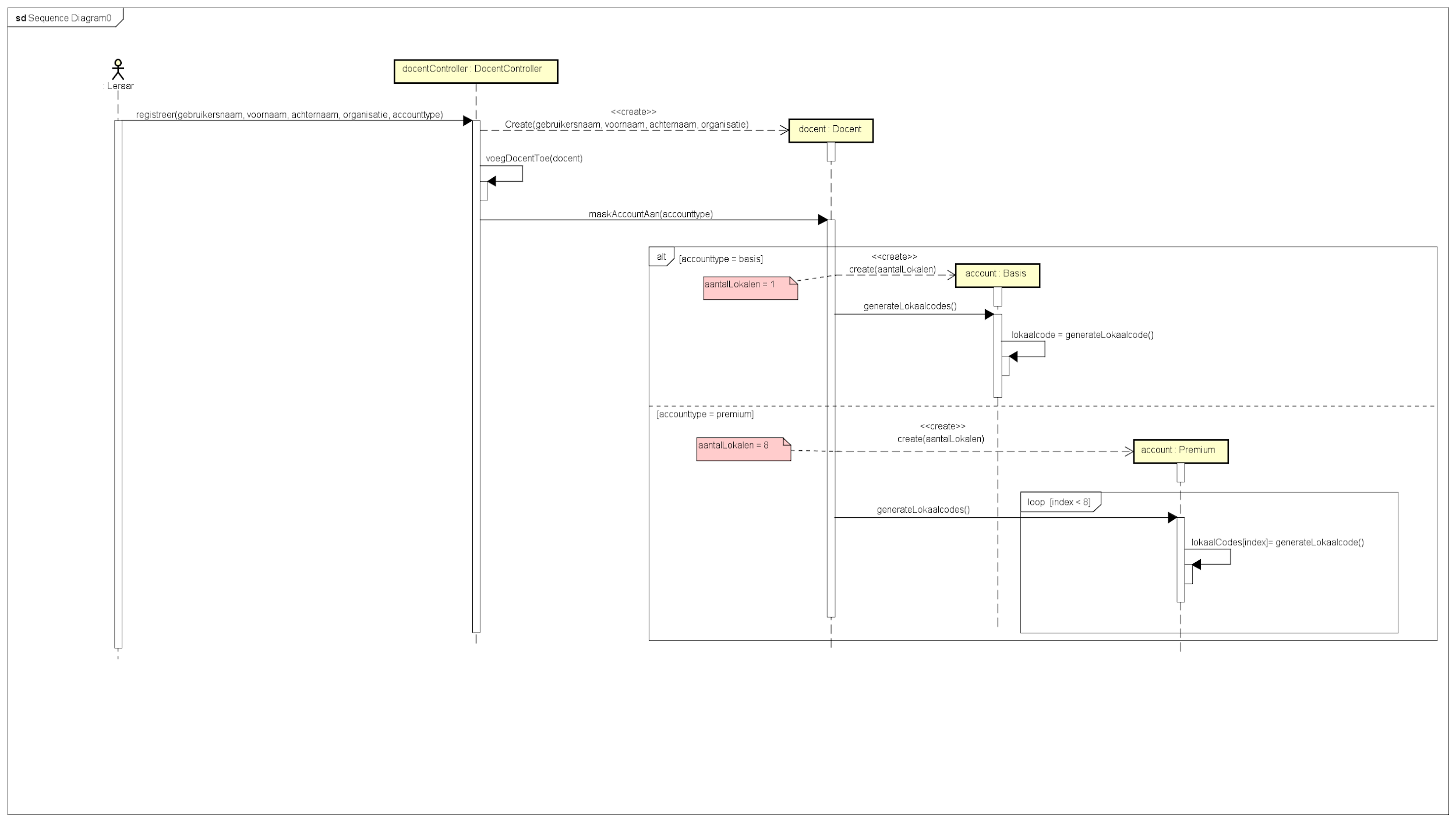
Als eerst komt het maken van de kennistoets aan bij de docentcontroller. Deze haalt de juiste docent op, op basis van de gebruikersnaam en delegeert het maken van de kennistoets door naar de docent zelf. Deze beheert namelijk de kennistoetsen.

Vervolgens wordt er gekeken welk type vraag de docent heeft gekozen en op basis hiervan de juiste vraag aangemaakt. De docent maakt de vraag aan en delegeert het creëren van de antwoorden door naar de vraag zelf. Als de complete vraag is aangemaakt, wordt deze toegevoegd aan de toets. De kennistoets wordt aan het docent object toegevoegd wanneer deze klaar is met het maken van de vragen.

Door de taken en verantwoordelijkheden zo goed mogelijk te verdelen wordt de koppeling zo laag mogelijk gehouden door elk klasse de juiste objecten te laten aanmaken en waar mogelijk taken door te laten delegeren.

### 

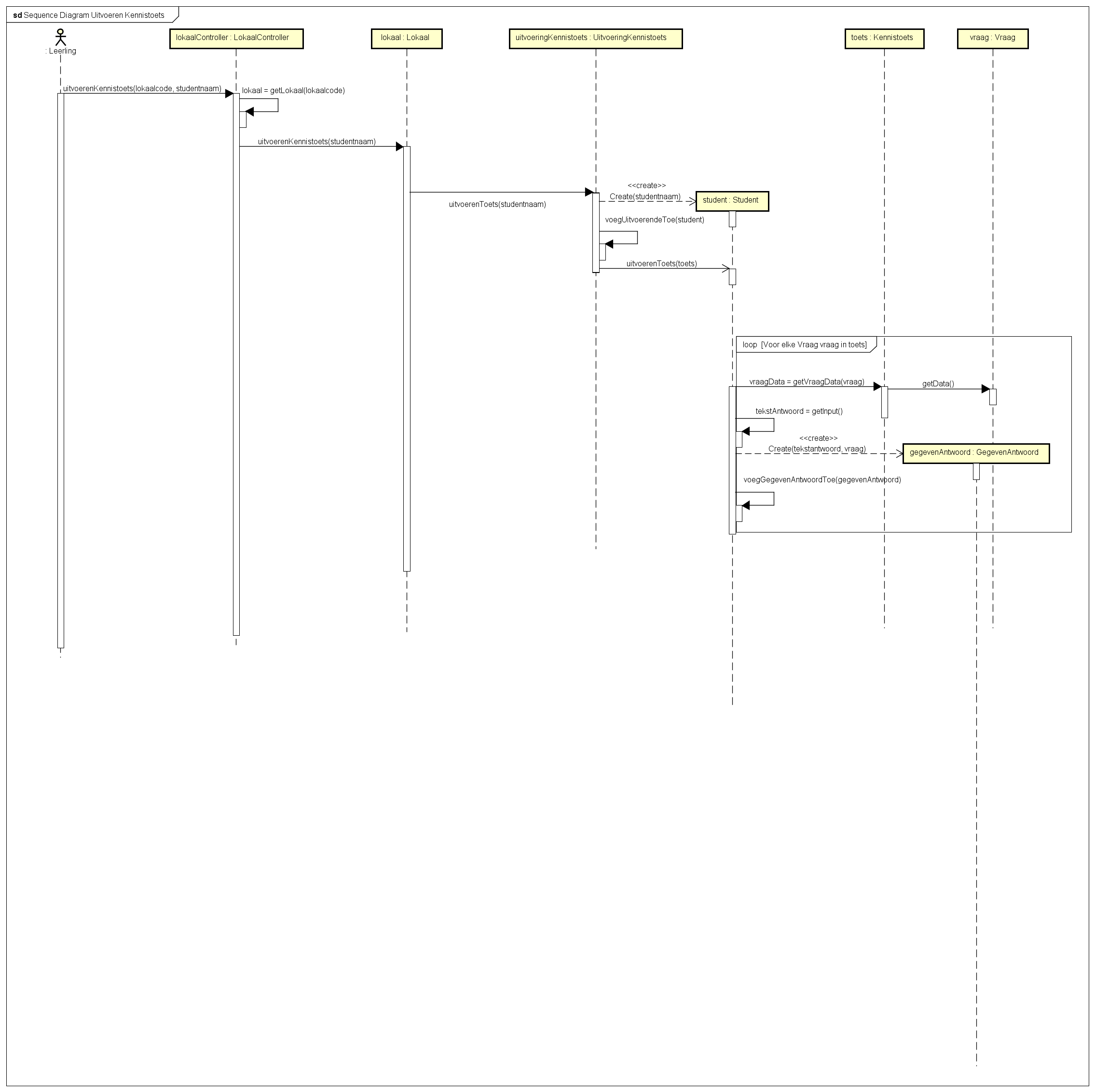
### 3.2.4 Registeren

[](Sequence%20Diagrammen/Registreren.png)

Figuur 4 Sequence diagram registreren

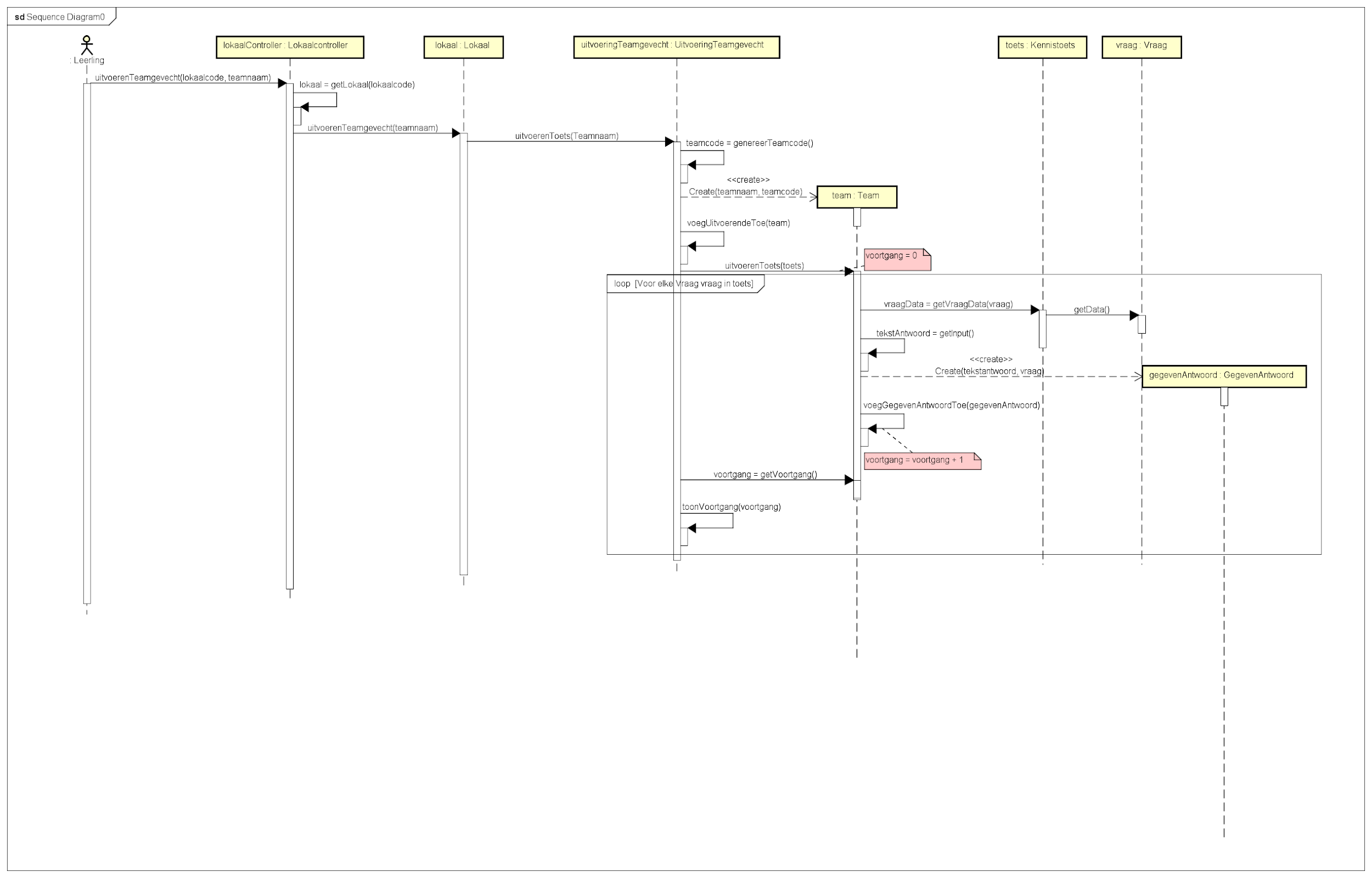
Bij het registreren wordt wederom het creator pattern toegepast. Als eerst maakt de docentcontroller een docent aan. Deze beheert namelijk de docent en krijgt via de system call van de actor alle benodigde informatie om een docent aan te maken. Als de docent is aangemaakt wordt het aanmaken van een account door gedelegeerd naar de docent zelf. De docent beheert namelijk zijn eigen account en maakt gebruik van zijn account, dus maakt hij zelf zijn account aan in plaats van de controller. Bovendien houdt dit de koppeling tussen de klassen laag, aangezien het aanmaken van het account door de docent controller een extra afhankelijkheid betekent.

### 3.2.5 Uitvoeren Kennistoets

[](Sequence%20Diagrammen/Uitvoeren%20kennistoets.jpg)

Figuur 5 Sequence diagram uitvoeren kennistoets

### 3.2.6 Uitvoeren Teamgevecht

[](Sequence%20Diagrammen/UitvoerenTeamgevecht.png)

Figuur Sequence diagram uitvoeren teamgevecht

### 

#### 

## Activity and State Diagrams

Voor het opgestelde model van Odinido is geen State Diagram gerealiseerd. Het enige object wat in het huidige model van state verandert is de uitvoering van de kennistoets. Deze kan verkerend in states als actief, non-actief en score berekenend. Deze states en de verandering er tussen zijn echter voor de hand liggend, vrij gemakkelijk te begrijpen en hebben nauwelijks impact op de werking van het systeem. Daarom is het niet relevant en essentieel om voor de verandering bij de uitvoering van de kennistoets een geheel state diagram op te stellen.

## Design decisions made for the sub-system

<Describe all design decisions made for the sub-system. Provide at least decision descriptions for all frameworks, libraries and other technologies used. Other decisions may be related to software patterns, system-structure, adapted principles or the like.> Beschrijven design beslissingen voor hele system hier waarschijnlijk GRASP en GOF patterns, maar kan ook al bij sequence diagrammen (maybe)

# Bijlage

[Link naar de afbeelding van het Design Class Diagram](Design%20Class%20Diagram/Design-class%20diagram.jpg)