

**Object Georiënteerd Ontwerp**

**Zelfevaluatie App**

Andaar Anwar

Arnalsteen Sam

Broeckhoven Mathieu

2TIVT

24/05/2019

University Colleges Leuven-Limburg  
Campus Proximus  
Geldenaaksebaan 335  
3001 Heverlee  
Tel. +32 (0) 16 375 700

Inhoud

[Algemene opmerking 3](#_Toc499494737)

[Vereisten 3](#_Toc499494738)

[Design patterns 5](#_Toc499494739)

[Speciale topics 7](#_Toc499494740)

[Werkverdeling 8](#_Toc499494741)

[Klassen diagram 9](#_Toc499494742)

# Algemene opmerking

Dit verslag gaat over de opdracht ‘Zelfevaluatie app’, dat zal meetellen voor 5 punten van je totaalscore van dit vak. Dit verslag vormt de toelichting bij jouw code.

**Je wordt geacht om je voor het verslag volledig aan deze template te houden. Buiten deze ‘algemene opmerking’ zijn alle paragrafen verplichte onderdelen van het verslag. Je print dit verslag af (dubbelzijdig) – 1 exemplaar per groep en levert dit in bij het einde van het laatste labo OOO.**

**Je maakt een zip file van al je source code (.java bestanden, geen .class bestanden) en van alle bestanden die van belang zijn voor deze OOO opdracht. Je voegt tevens de laatste versie van je verslag (Word document) toe aan deze zip file.**

**Je laadt deze zip file en jar file op via Toledo uiterlijk op vrijdag 31/05/2019 om 23.59 uur. Naam van de zip file: Zelfevaluatie\_ gevolgd door de familienamen van de groepsleden gescheiden door underscore (bv. Zelfevaluatie\_Dox\_Lenaerts\_Jansen.zip).**

**Het spreekt voor zich dat de code op Toledo overeenstemt met de code in je repository op GitHub.**

# URL

<https://github.com/AnwarHa/Arnalsteen_Broeckhoven_Haidari_rep2_2019>

# Vereisten

Het doel van dit project was dat we een app maakten waar

Geef voor het gehele project aan welke vereisten (evt. zelf verder uitgewerkt/opgesplitst aan de hand van de opgave) je succesvol hebben geïmplementeerd, en welke topics niet gelukt zijn. Indien je een deel van de voorziene vereisten niet afgewerkt hebt, geef dan aan waarom niet… De reden kan “tijdsgebrek” zijn, het kan een issue zijn “wist niet hoe, het crashte”, of het kan zijn dat je een zeer goede reden had om het niet te implementeren…

Voeg het definitieve gegenereerde klassendiagramma van je code toe, als afzonderlijke afbeelding, als bijlage bij dit rapport.

Als er zaken zijn uit de opdracht die je niet hebt kunnen uitwerken of die je beter zou willen uitwerken, dan som je deze hier tevens op (in laatste rij van tabel).

|  |  |  |
| --- | --- | --- |
| User story | OK? | indien niet ok – wat lukt niet (zie acceptatie-criteria) en waarom? |
| 1. Overzicht categorieën tonen | OK |  |
| 1. Nieuwe categorie toevoegen | OK |  |
| 1. Overzicht vragen tonen | OK |  |
| 1. Nieuwe vraag toevoegen | OK |  |
| 1. Test afleggen | OK |  |
| 1. Resultaat tonen met punten | OK |  |
| 1. Resultaat tonen met feedback | OK |  |
| 1. Type feedback kiezen | OK |  |
| 1. Resultaat tonen zonder fouten | OK |  |
| 1. Test reeds afgelegd | OK |  |
| 1. Test opnieuw afleggen | OK |  |
| 1. Bestaande categorie aanpassen | OK |  |
| 1. Bestaande vraag aanpassen | OK |  |
| 1. Test selecteren uit lijst van excel bestanden |  |  |

# Design patterns

Geef voor elk gezien patroon aan waar je het gebruikt hebt (mogelijkerwijs meer dan eens). Genereer een klassendiagram voor elke situatie waarin je het patroon gebruikt hebt.

Geef extra informatie (voordelen / waarom / …). Als je een patroon niet toegepast hebt, leg uit waarom niet.

|  |  |  |
| --- | --- | --- |
|  | Toegepast (ja/nee)  In welke stories(nr)  Waarom toegepast(voordeel) | Bijhorend klassendiagram (gegenereerd vanuit je java code) |
| Observer | Ja, story 3 en 4 | We hebben het observer patroon gebruikt om de antwoorden op een vraag toe te kunnen voegen. We wilden meteen als we op ‘add’ duwden kunnen zien welk de mogelijke antwoorden waren op een bepaalde vraag. Ook om antwoorden te verwijderen werd het observer patroon toegepast. Tot slot om de overviewpanes up-to-date te houden (als er een nieuwe categorie of vraag wordt toegevoegd). |
| Strategy | ja, story 1,2,3,4 | Wordt gebruikt in de TextDatabaseStrategy en de RamDatabaseStrategy  Gebruikt in de TextDatabaseStrategy en ramDatabaseStrategy  Om te kunnen kiezen welke manier van opslagen we gebruiken, meerdere manieren mogelijk. De implementatie is zeer gelijkaardig. Het wisselen van Strategy is zeer eenvoudig. |
| Simple Factory | ja | simpleDatabaseFactorySingleton: klasse die nieuwe objecten van de database kan aanmaken. |
| Façade | ja, | databaseService, om de logica van de database gescheiden te houden van de rest van de applicatie |
| Singleton | ja | simpleDatabaseFactorySingleton: om 1 instatie van een databaseStrategy aan te maken. |
| State | ja, maar niet volgens patroon | Gebruikt in “evaluaton.properties” en in “PropertiesDatabase”: om te checken of de test al is afgelegd en om de score van de vorige test te tonen. |
| MVC | Ja, allemaal | We hebben volgens het MVC model gewerkt om de belangen van klassen niet onderling te mengen maar een gecontroleerde logica te creëren. |
| Decorator | nee | Overbodig in deze applicatie. |
| Template method | nee | We zijn er niet in geslaagd om dit patroon foutloos te implementeren. |

# Speciale topics

Geef voor elke “speciale topic” aan of je het gebruikt hebt of niet, en zo ja, waar. Toon eventueel aan met een klassendiagram.

|  |  |  |
| --- | --- | --- |
|  | Toegepast (ja/nee)  In welke stories(nr)  Waarom toegepast(voordeel) | Bijhorend klassendiagram (uit object aid) of extra uitleg |
| Enum | ja | DatabaseStrategies, bevat verschillende strategieën voor de database. Dit is nodig voor met reflections verschillende objecten aan te maken. |
| Properties | ja | evaluation.properties: staat hoe de gegevens worden opgeslagen, of de test reeds is afgelegd en de score van de laatste test |
| Reflection | Ja | zit in simpleDatabaseFactorySingleton: om objecten te maken op basis van de klassennaam: bv “database.” + gewensteStrategy + “category” |
| Andere… |  |  |

# Werkverdeling

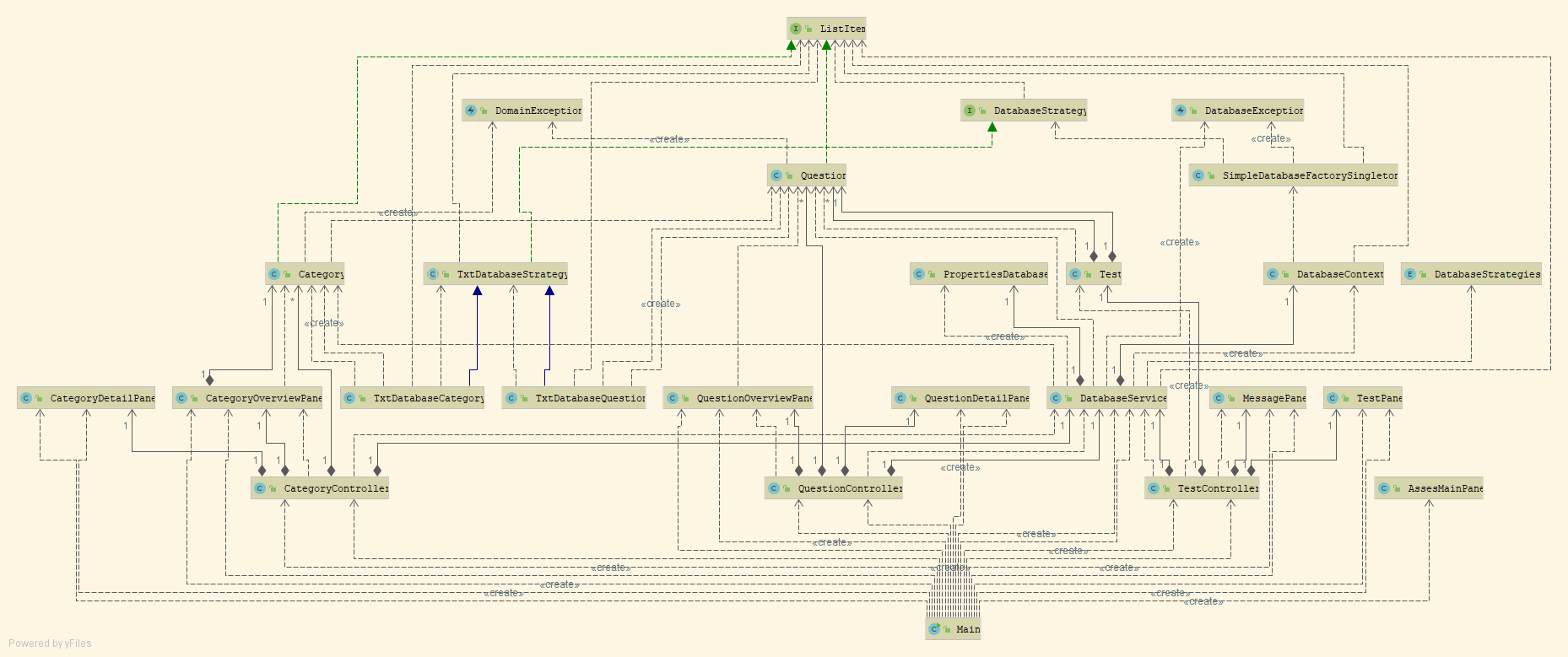
Geef aan in percentages hoeveel je bij benadering gespendeerd hebt aan deze opdracht.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Mathieu | Anwar | Sam | Totaal |
| Ontwerp | 40 | 40 | 20 | 100% |
| Klassendiagrammen | 33 | 33 | 33 | 100% |
| Implementatie | 35 | 35 | 30 | 100% |
| Verslag | 15 | 15 | 70 | 100% |

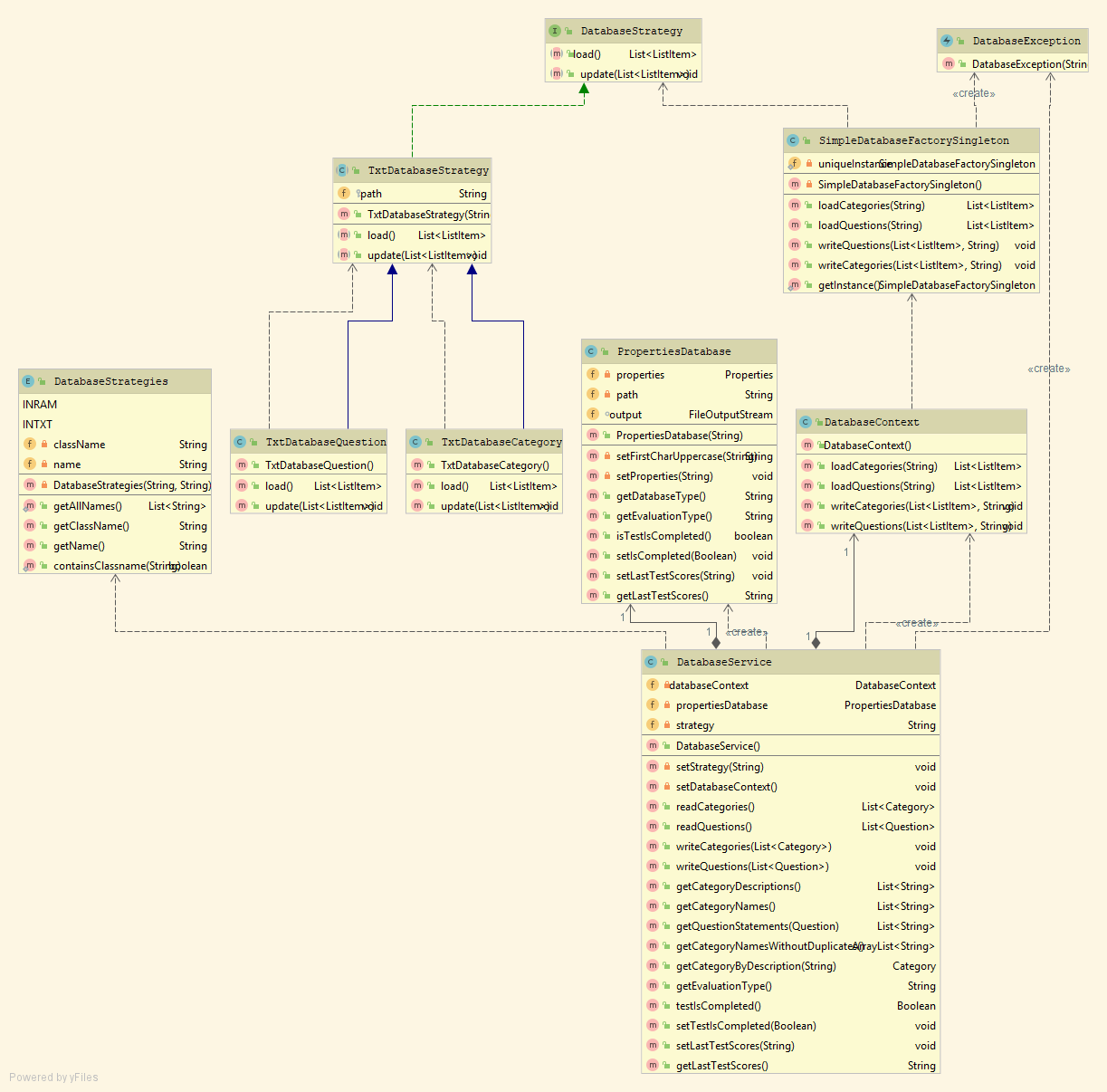
# Klassen diagramma’s

Voeg het definitieve gegenereerde klassendiagram van je code toe), als afzonderlijk afbeelding-bestand. **Dit klassendiagram moet goed leesbaar zijn.** Spreid het over meerdere pagina’s (bv. 1 blad per package (MVC) en 1 overzicht klassendiagram (zonder attributen en methoden).

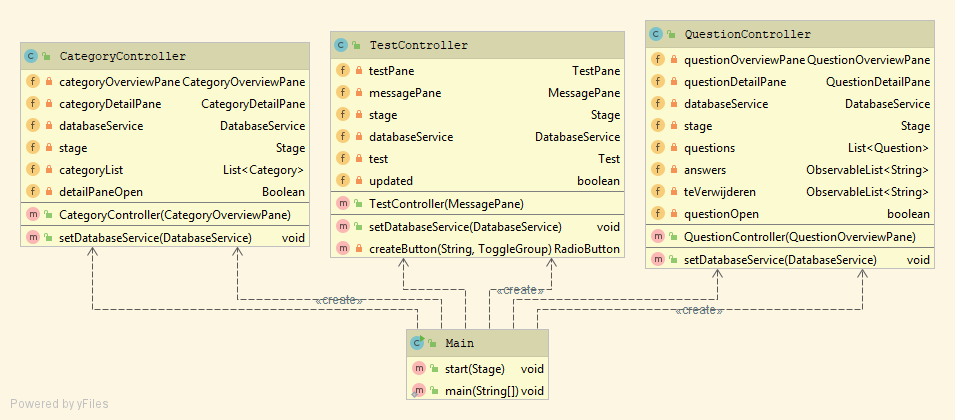
Project:



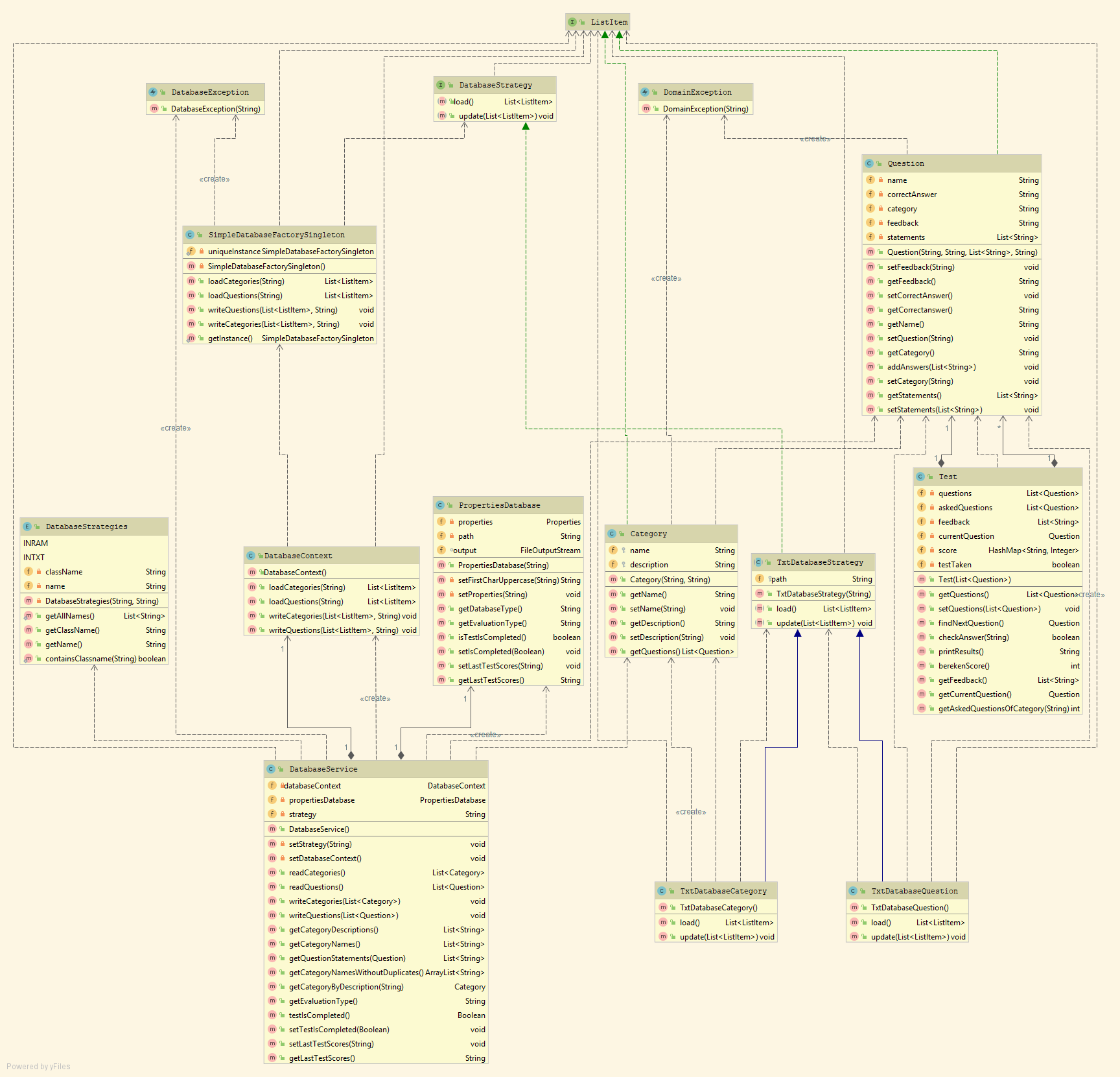
Database:

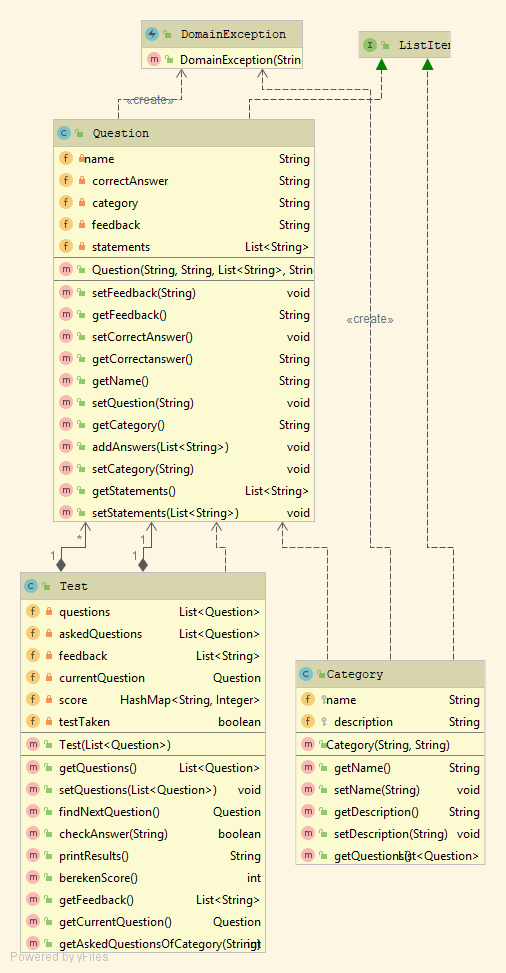


Main & controllers:



Model en Database



Model klassen:

View & controller

