# Practicumopdracht Hibernate en JSF : Kwekkeraar (versie C)

## Opdracht

De neefjes van Donald Duck, Kwik, Kwek en Kwak, kwekken wat af. Kwak kwekt het meest, maar Kwek is het kwikst met het kwakken van zijn kweks op het net. Na de kwek van Kwek, kwakt Kwik zijn kwek op het net. Daarna volgt Kwak met zijn kwek, omdat hij niet zo kwik is.

Donald wil het gekwekker volgen, maar heeft wat moeite met het kwikke tempo. Een mogelijkheid zou tweeten op Twitter zijn, maar hij vindt dat er teveel getoeterd en getetterd wordt en wil een beschermde overzichtelijke omgeving aanbieden waarin hij een belangrijke rol speelt.

Gevraagd wordt om uiteindelijk met behulp van JSF een variant van Twitter, de Kwekkeraar, te ontwikkelen die aan de volgende eisen voldoet:

* Er is een server waarop de kweks worden bewaard. CHECK
* Iedereen kan een accountant met een wachtwoord aanmaken, waarna hij zijn kweks op de Kwekkeraar kan kwakken. CHECK
* Een kwek bestaat uit maximaal 255 karakters. CHECK
* Er is een lijst met namen van alle kwekkeraars. CHECK  
  Een gebruiker (ook een kwekkeraar) kan een naam van deze lijst selecteren en de kweks van deze kwekkeraar lezen. CHECK  
  Deze lijst wordt na het aanmaken of verwijderen van een account runtime niet geupdate.

CHECK

* Een gebruiker kan een lijst aanleggen met zijn favoriete kwekkeraars. Elke keer als hij inlogt,wordt zijn lijst met favoriete kwekkeraars getoond.

TODO

* Donald kan als beheerder alle kweks lezen en gebruikers het recht tot het kwakken ontnemen en teruggeven.
* Een kwekkeraar kan zijn account verwijderen. Daarmee worden ook al zijn kweks verwijderd. Wie kweks wil lezen van een verwijderde kwekkeraar krijgt een passende foutboodschap. CHECK

TODO

* Een kwekkeraar kan na het kwakken van tenminste 200 kweks zijn account opkrikken tot een platina account. Hierdoor kan de kwekkeraar aan een kwek een *I\_kwaik\_it* (het equivalent van I like it) toe te voegen. Hierdoor wordt de naam van de kwekkeraar naast/onder de kwek vermeld.

## Tips en eisen voor de aanpak

* Maak voor je begint eerst een analyse-klassendiagram van de opdracht. Het is verstandig om deze aan de practicumdocent te laten zien, zodat je weet of je op de goede weg bent.
* Breid het analyse-klassendiagram uit met navigaties (van associaties) en types. Hierdoor ontstaat eigenlijk een design klassendiagram. Het design klassendiagram zal misschien nog niet helemaal compleet zijn omdat je later allicht nog methodes wilt aanvullen.
* Maak a.h.v. het klassendiagram een database en bijhorende model klassen.
* Maak mapping files om de tabellen en klassen aan elkaar te koppelen.
* Maak verschillende unit-tests waarmee je het volgende kan:
  + Van iedere klasse objecten opslaan in de database
  + Van iedere tabel records uit de database halen en vertalen naar objecten.
  + Van iedere klasse objecten updaten in de database.

Het bovenstaande moet met Hibernate worden gerealiseerd. Als je dit hebt volbracht, dan kun je dit gedeelte af laten tekenen bij de practicumdocent.

* Vervolgens kun je het JSF gedeelte maken. Je gebruikt hierbij de model klassen en hibernate om de model objecten op te slaan. Je bent in principe vrij in de layout en schermindeling die je gaat gebruiken. Het is echter wel aan te raden om van te voren een schermindeling te maken en om na te gaan of nog wel aan alle eisen wordt voldaan.