Kwetter Ontwerp

# User stories

## Stakeholders

Kwetter

## Rollen

Gebruiker

Moderator

Administrator

## User stories

### Profiel pagina

1. Als Kwetter, wil ik dat gebruikers hun eigen profiel foto, naam (moet uniek zijn) en details kunnen wijzigen, zodat ze zich goed kunnen weergeven op kwetter en zich vertrouwd kunnen voelen.
2. Als Kwetter, wil ik dat gebruikers andere profielen kunnen bekijken, zodat ze kunnen zien met wie ze communiceren en zich vertrouwd kunnen voelen.
3. Als Kwetter, wil ik dat gebruikers de laatste tien tweets kunnen zien van een gebruiker, zodat ze snel op de hoogte kunnen zijn van de laatste tweets van een gebruiker en op de hoogte kunnen blijven.
4. Als Kwetter, wil ik dat gebruikers kunnen zien welke gebruikers een gebruiker volgen, zodat ze weten wie elkaar volgt en of er nieuwe connecties kunnen ontstaan.
5. Als Kwetter, wil ik dat gebruikers kunnen zien welke gebruikers door een gebruiker gevolgd wordt, zodat te zien is wie elkaar volgt en of er nieuwe connecties kunnen ontstaan.
6. Als Kwetter, wil ik dat gebruikers een hartje kunnen geven aan een kweet, zodat gebruikers worden aangemoedigd meer kweets te schrijven.

### Startpagina

1. Als Kwetter, wil ik dat gebruikers een kweet kunnen zoeken, zodat ze een kweet opnieuw kunnen lezen.
2. Als Kwetter, wil ik dat gebruikers een kweet kunnen maken, zodat deze gelezen kan worden door andere gebruikers.
3. Als Kwetter, wil ik dat gebruikers een tijdlijn te zien krijgen met daarin Kweets van zich zelf en van de personen die ze volgen, zodat ze op de hoogte kunnen blijven en aangemoedigd worden om zelf ook een kweet te maken.
4. Als Kwetter, wil ik dat gebruikers kunnen in- en uitloggen, zodat ze veilig gebruik kunnen maken van hun accounts.

### Registreren en inloggen

1. Als Kwetter, wil ik dat een gebruikerspagina is gepersonaliseerd, zodat ze vertrouwd kunnen raken met de applicatie.
2. Als Kwetter, wil ik dat gebruikers zich authentiseren om toegang te krijgen tot wijzigbare of gepersonaliseerde pagina’s, zodat gebruikers zich veilig voelen op dergelijke pagina’s.
3. Als Kwetter, wil ik dat gebruikers zichzelf kunnen registreren, zodat ze weten dat dit op een veilige manier gebeurt en hun persoonlijke gegevens veilig zijn.

### Beheer

1. Als Kwetter, wil ik dat bepaalde gebruikers een moderator rol kunnen krijgen, zodat deze moderators de applicatie veilig en vertrouwd kunnen houden.
2. Als Kwetter, wil ik dat moderators een lijst van kweets kunnen opvragen, zodat kweets met ‘onheus taalgebruik’ meteen opvalt die de applicatie onveilig beïnvloeden.
3. Als Kwetter, wil ik dat moderators een kweet met ‘onheus taalgebruik’ die de applicatie onveilig beïnvloedt kan verwijderen, zodat de applicatie weer veilig en vertrouwd wordt.
4. Als Kwetter, wil ik dat moderators en administrators een lijst van gebruikers met rollen kunnen opvragen, zodat te zien is wie de applicatie gebruikt en op die manier duidelijk wordt of de applicatie nog veilig is.
5. Als Kwetter, wil ik dat een administrator extra rollen kan toevoegen, zodat deze onderhoud kunnen plegen om de applicatie veilig te houden.
6. Als Kwetter, wil ik dat een administrator een rol kan toewijzen aan een gebruiker, zodat deze mee kan helpen de applicatie veilig te houden.

# Domeinmodel

Het domeinmodel is als apart bestand ‘Class Diagram.vsdx’ bijgevoegd.

# Specificatie van de servicelaag

## UserService

*/\*\*  
 \* Creates a user of Kwetter in the system.  
 \*  
 \** ***@param*** *user the user that will be created  
 \*/*public void createUser(User user) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Updates a user of Kwetter in the system.  
 \*  
 \** ***@param*** *user the updated user  
 \*/*public void updateUser(User user) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Searches for a user with the given name  
 \* Returns null if no user can be found  
 \*  
 \** ***@param*** *name the name of the requested user  
 \** ***@return*** *the found user based on the given name  
 \*/*public User getUser(String name) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Searches for a user with the given id  
 \* Returns null if no user can be found  
 \*  
 \** ***@param*** *id the id of the requested user  
 \** ***@return*** *the found user based on the given id  
 \*/*public User getUser(int id) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Makes a relation where one user follows another  
 \*  
 \** ***@param*** *follower the id of the user that is the follower  
 \** ***@param*** *following the id of the user that gets followed  
 \*/*public void follow(int follower, int following) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Returns a list of users that follow the given user  
 \*  
 \** ***@param*** *following the id of the user that is being followed  
 \** ***@return*** *the list of followers  
 \*/*public List<User> getFollowers(int following) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Returns a list of users that the given user follows  
 \*  
 \** ***@param*** *follower the id of the user that follows  
 \** ***@return*** *the list of following users  
 \*/*public List<User> getFollowing(int follower) {  
 throw new NotImplementedException();  
}

## KweetService

*/\*\*  
 \* Creates a new kweet in the system.  
 \*  
 \** ***@param*** *kweet to be created.  
 \*/*public void createKweet(Kweet kweet) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Makes the given user like the given kweet.  
 \*  
 \** ***@param*** *kweet the kweet to be liked  
 \** ***@param*** *userId the id of the user that likes the kweet  
 \*/*public void likeKweet(Kweet kweet, int userId) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Gives a list of kweets of the given user itself  
 \* and of the users that the given user follows  
 \*  
 \** ***@param*** *userId the id of the user where the timeline  
 \* will be generated for  
 \** ***@return*** *the generated timeline based on a list of kweets  
 \*/*public List<Kweet> getTimeline(int userId) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Gives a list of the latest kweets of the given user  
 \* The list is limited to the amount  
 \* If the amount is smaller or equal to 0, the list is unlimited  
 \*  
 \** ***@param*** *userId the id of the user where the latest kweets are requested for  
 \** ***@param*** *amount the amount of kweets that are requested  
 \** ***@return*** *the latest amount kweets of the given user  
 \*/*public List<Kweet> latestKweets(int userId, int amount) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Gives a list of kweets that match the given keyword  
 \*   
 \** ***@param*** *keyword the word that must be in the message of the kweets  
 \** ***@return*** *the kweets where the message match the keyword  
 \*/*public List<Kweet> searchKweet(String keyword){  
 throw new NotImplementedException();  
}

## ModService

*/\*\*  
 \* Lets a mod or admin to change the role of a user  
 \*  
 \** ***@param*** *userId the id of the user where the role will be changed of  
 \** ***@param*** *role the role the user get  
 \*/*public void changeRole(int userId, String role) {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Gives a list of all users with its roles for a mod or admin  
 \*  
 \** ***@return*** *the list of all users with its roles  
 \*/*public List<User> getAllUsers() {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Gives a list of all kweets for a mod or admin  
 \*  
 \** ***@return*** *the list of all kweets  
 \*/*public List<Kweet> getAllKweets() {  
 throw new NotImplementedException();  
}  
  
*/\*\*  
 \* Lets a mod delete a kweet  
 \*   
 \** ***@param*** *kweet the kweet that has to be deleted  
 \*/*public void deleteKweet(Kweet kweet){  
 throw new NotImplementedException();  
}