

**Rapport gegevens documenteren en archiveren**

Januari 2020

# 

# **Inhoudsopgave**

[**Inhoudsopgave**](#_t4ebjjtroxpo) **2**

[**Inleiding**](#_kcgtui27zbj2) **3**

[**Beschrijving van de database**](#_e0u8nlwvtsto) **3**

[ERD](#_dmks0k73j5a3) 3

[**Procedures met betrekking tot opslag van gegevens**](#_dhtpx9b1jqxz) **4**

[Opslaglocatie](#_rspcjtcbfvxs) 4

[Autorisatie](#_op2he8hh6u5m) 4

[Use cases gebruiker](#_ypjfga41fg6i) 4

[Use cases admin](#_4n3lx3u2o8vc) 5

[Use case systeem](#_tn3pb8wwrhh1) 6

[Backup-procedures](#_qnntaf6ve6f) 6

[**Controle van bestaande documentatie**](#_654wk2lof9sz) **6**

[**Procedure met betrekking tot het bijwerken van systeemdocumentatie (FO/ TO)**](#_cppxj8wwonpc) **7**

# 

# **Inleiding**

Dit Rapport gegevens documenteren en archiveren bevat informatie over het onderhouden van de documentatie van het project. Ook wordt gedocumenteerd hoe andere data wordt opgeslagen M.B.T het project. Het project waar ik het over heb is het project de Update Tracker versie definitief.

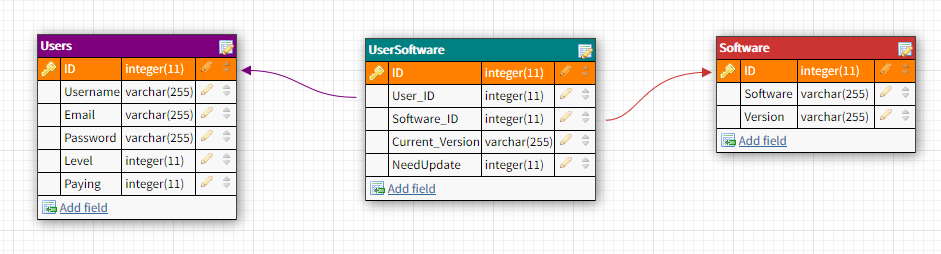
# 

# **Beschrijving van de database**

## **ERD**

Hieronder is een afbeelding die het databasemodel beschrijft. Het laat de relaties zien van de database. Ook staat er bij wat voor data type het is. Er is te zien dat User\_ID gekoppeld is met ID van de tabel Users. Dit schema ga ik gebruiken als ik de database ga maken. Onder de afbeelding staat een tabel wat meer uitleg geeft over de tabel.

De tabel UserSoftware bevat 2 sleutels uit andere tabellen namelijk User\_ID die koppelt welke user een software heeft en Software\_ID die koppelt welke software een user heeft. De pijlen geven die relaties aan.



## 

# 

# **Procedures met betrekking tot opslag van gegevens**

## **Opslaglocatie**

Alle documenten worden geupload naar [github](https://github.com/Jurjen-V/Update-Tracker) door mij zelf. Op github staan alle documenten openbaar en beschikbaar voor iedereen.

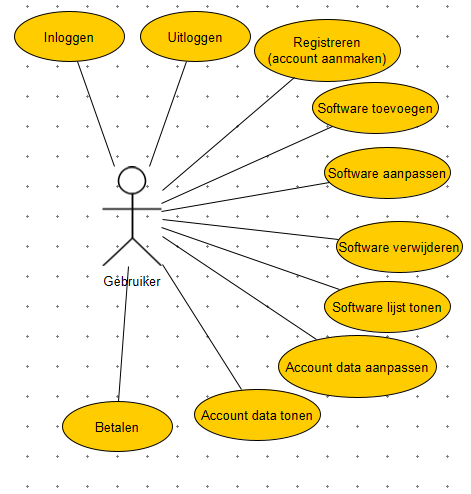
## **Autorisatie**

Het systeem kent meerdere rollen. Zo is er een admin, gebruiker en het systeem zelf. Alle drie hebben zij hun eigen unieke functionaliteiten en doeleinden. Hieronder staan van alle drie use cases. Zo is het duidelijk wie wat wel en niet mag.

### 

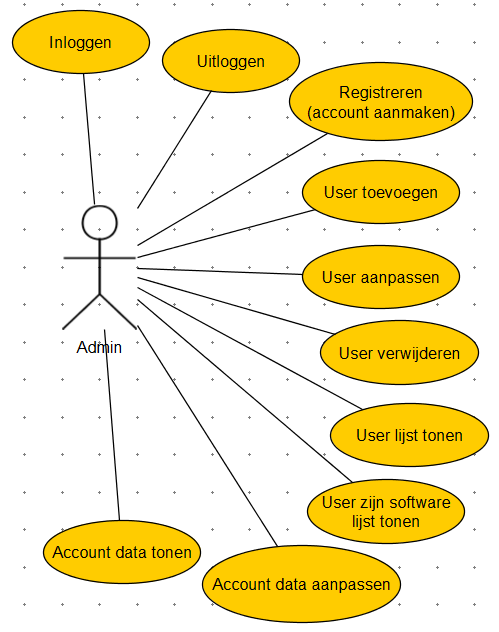
### Use cases gebruiker

In deze use case is omschreven welke functionaliteiten de normale gebruiker heeft. De gebruiker heeft toegang tot inloggen, uitloggen, registreren (account aanmaken), software toevoegen, software aanpassen, software verwijderen, software lijst tonen, account data aanpassen, account data tonen en betalen.



### Use cases admin

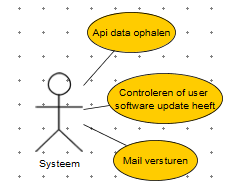
In deze use case is omschreven welke functionaliteiten de admin heeft. De admin heeft toegang tot inloggen, uitloggen, registreren (account aanmaken), user toevoegen, user aanpassen, user verwijderen, user lijst tonen, user zijn software lijst tonen, account data aanpassen en account data tonen.



### 

### Use case systeem

In deze use case is omschreven welke functionaliteiten het systeem heeft. Het systeem runt de volgende functionaliteiten automatisch; Api data ophalen, controleren of user software update heeft en mail versturen.



## **Backup-procedures**

Backup van code en documentatie staat online op [github](https://github.com/Jurjen-V/Update-Tracker). De verantwoordelijkheid van de database backup wordt uitbesteed aan de hosting.

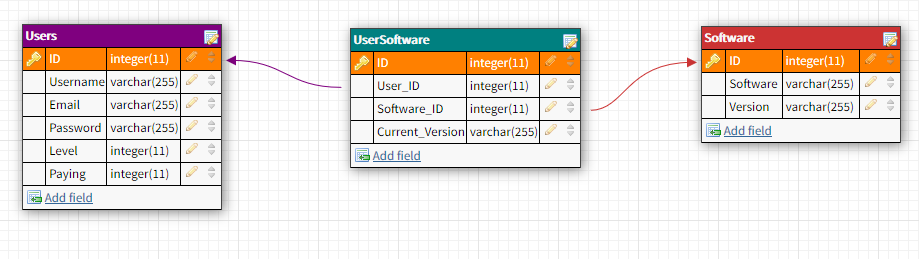
# 

# 

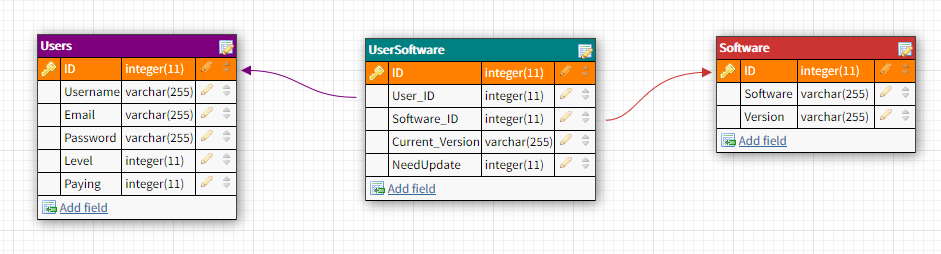
# **Controle van bestaande documentatie**

Het Technisch ontwerp is niet helemaal accuraat meer. Het database model is namelijk een klein beetje veranderd in de loop van het ontwikkelen.

Oude database model:



Nieuwe database model:



# 

# **Procedure met betrekking tot het bijwerken van systeemdocumentatie (FO/ TO)**

Als het FO en of TO niet meer correct zijn worden ze bijgewerkt zodat ze weer aansluiten bij het project. De aanpassingen worden geupload naar [github](https://github.com/Jurjen-V/Update-Tracker). De nieuwste versie is daar dus altijd te vinden.