Timothy Vanhee

Technisch document smart shower

# Voorwoord

Dit document is een onderdeel van het project SmartShower. Smart Shower is een project bestaande uit vier Howest studenten waarvan 1 3IPO-student en 3 2NMCT-studenten.

Opstelling

Klant: IPO-student

Developmentteam:

* Céline Abbeloos
* Timothy Vanhee
* Gillian Vanhaverbeke

Dit technisch document is bedoeld voor zij die graag meer kennis hadden verworven over de ontwikkeling en setup van de applicatie.

# Inhoudsopgave

[Voorwoord 1](#_Toc505117318)

[Inhoudsopgave 2](#_Toc505117319)

[Benodigheden 4](#_Toc505117320)

[Hardware 4](#_Toc505117321)

[electronica 4](#_Toc505117322)

[behulzing 5](#_Toc505117323)

[Extra handige benodigheden 5](#_Toc505117324)

[Software 5](#_Toc505117325)

[Setup 6](#_Toc505117326)

[Configuratie Aruino (UNO) 6](#_Toc505117327)

[Mogelijke problemen 6](#_Toc505117328)

[Configuratie raspberry pi 3 7](#_Toc505117329)

[mogelijke problemen 8](#_Toc505117330)

[smartShower API 11](#_Toc505117331)

[languages, libraries, tools and technologies used 11](#_Toc505117332)

[Overzicht van de API 12](#_Toc505117333)

[klassen 16](#_Toc505117334)

[Configuratie 16](#_Toc505117335)

[Databases 17](#_Toc505117336)

[MSSQL: SmartShowerDb 17](#_Toc505117337)

[ERD-schema 17](#_Toc505117338)

[Tabellen 18](#_Toc505117339)

[Overzicht gebruikte queries 19](#_Toc505117340)

[CosmosDb smartShower 21](#_Toc505117341)

[Aanmaak van de database 21](#_Toc505117342)

[Overzicht gebruikte queries 22](#_Toc505117343)

[SmartShowerApp Cross-Platform (Xamarin.Forms) 23](#_Toc505117344)

[languages, libraries, tools and technologies used 23](#_Toc505117345)

[Extra nuget packages 23](#_Toc505117346)

[Lijst extra geinstalleerde packages 24](#_Toc505117347)

[Structuur PCL project 25](#_Toc505117348)

[Assets 26](#_Toc505117349)

[Model 26](#_Toc505117350)

[Conclusies, opmerkingen en suggesties 28](#_Toc505117351)

[RTC module Arduino 28](#_Toc505117352)

[RFID 28](#_Toc505117353)

# Benodigheden

Voor de kostprijs verwijs ik naar de bom die bij dit document meegeleverd werd.

## Hardware

### electronica

#### Led die water simuleert

Voor de ontwikkeling van het prototype SmartShower is, in plaats van water, een led voorzien ter aanduiding wanneer er al dan niet water uit de douchekop stroomt. De reden hiervoor is de hoge kostprijs van een industriele waterpomp. Ook was er tijdens de bouw van het project geen rekening gehouden met de fysica der over- en onder-druk.

#### Arduino UNO

Een simpele Arduino UNO (mag ook andere uitvoering zijn) volstaat om de data van de sensoren uit te lezen.

#### Potentiometer voor Water

Bij het prototype wordt gebruik gemaakt van een potentiometer om het waterverbruik te bepalen. De bedoeling is om bij een prototype die werkt op water, dat de potentiometer vervangen wordt door een effectieve waterkraan. De berekingen voor het water kunnen als volgt worden geimplementeerd: de stand van de waterkraan bepaald de druk op het water. Hoe hoger de druk, hoe meer water er door de kraan vloeit. Deze waarde x aantalSecondenPerSessie = totaalWaterVerbruik.

#### Rotary Encoder Temperatuur

Gezien de start temperatuur vooraf ingesteld wordt per gebruiker, heeft elke gebruiker een andere start waarde. Om de stand van de temperatuur manueel te verhogen/verlagen maken we gebruik van een Rotary Encoder en niet de alomwelbekende potentiometer. We hebben gekozen voor een Rotary Encoder omdat deze geen limiet kent.

**Noot**: wanneer men het project uitvoert met een werkende waterkkraan is een temperatuursensor voorzien. Gezien bij het prototype er geen water aanwezig is en er dus geen materie is om de temperatuur van te meten, is besloten dat de stand van de Rotary Encoder gelijkstaat aan de werkelijke watertemperatuur.

#### Rotary Encoder Profiel selecteren

Voor de gebruiker van start gaat met zijn douche, zal deze zijn profiel selecteren. Dit gebeurd aan de hand van een Rotary Encoder. Aan de Rotary Encoder hangt een RGBled die telkens een andere kleur weergeeft per profiel. Zo weet de gebruker wanneer zijn profiel geselecteerd is.

**Noot**: voor het prototype smartShower is hier gebruik gemaakt van een potentiometer in plaats van een rotarty encoder. Merk op dat een rotarty encoder gebruikersvriendelijker was geweest.

#### WS2812B 5050 RGB Led strip (1m, 30 leds) voor selectie profielen

Aan de douchekraan zelf hangt een ledstrip in cirkelvorm. Zo weet de gebruiker welk profiel geselecteerd is.

#### Raspbery Pi 3 Model B + Voedingadapter (output: +5V gelijkstroom 2A)

Dit model is de krachtigste in de reeks. Een Raspberry Pi 2 zou eventueel ook werken, maar dit is niet uitgetest.

#### USB kabel voor seriele communicatie

We hebben gekozen voor seriele communicatie tussen de Arduino en Raspberry Pi 3. Eerst was gedacht om data door te sturen via bluetooth maar de USBkabel lag in handbereik en hielp ons lichtjes de kostprijs van het project te dalen.

#### Jump-wire kabels M-M, M-F + (optioneel breadboard)

Kabels om de electronische schakeling te bouwen.

### behulzing

#### PLA Plastic

Bij het prototype is PLA materiaal gebruikt om de print van de douche uit te voeren. In productie omgeving zal wellicht gebruik worden gemaakt van een proffesionele douche die voorzien is om water te leveren.

### Extra handige benodigheden

om de ontwikkeling van de applicatie te kunnen voorzien:

#### Een windows machine

De software is ontwikkeld op een Windows 10 machine.

#### SD kaartlezer en een SD moederkaart

Dit hebben nodig om de raspberry pi correct te configureren

#### Een ethernet kabel

Voor toegang tot internet bij problemen. Alsook de ssh communcatie naar de raspberry Pi.

**noot**: voor een overzicht van de totale kostprijs verwijs ik u naar de bij dit document geleverde BOM.

## Software

De software van de applicatie is bij dit document bijgeleverd, gecompimeerd in smartShower.zip

Vanuit Visual Studio kan de applicatie op een android toestel worden geladen.

Voor de ontwikkeling van de software smartShower zijn volgende tools gebruikt:

* Visual Studio 2017 (Cross-Platform App Xamarin.forms & Azure Functions)
* Android SDK Tools
* Microsoft Azure
* Arduino IDE
* Microsoft SQL Server Management Studio 17
* PyCharm 2017.2.3
* SQL DOC4 (datagenerator voor testdata)

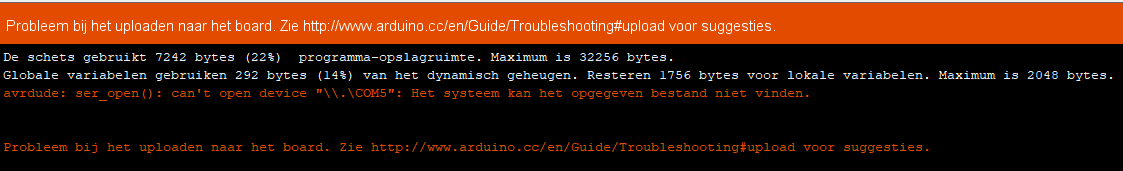
# Setup

## Configuratie Aruino (UNO)

De eerste stap is het dowloaden van de [Arduino Ide](https://www.arduino.cc/en/Main/Software). Vervolgens kan u de software installeren. Tijdens de installatie kunt u de arduino aansluiten via de usb met de computer. Eenmaal de installatie is voltooid kan u het arduino.ino[[1]](#footnote-1) bestand uploaden.

### Mogelijke problemen

#### Problemen bij het uploaden naar het board



Navigeer onder het tabblad hulpmiddelen >> poort en selecteer de aangeven poort bv COM3.

## Configuratie raspberry pi 3

Voor de Raspberry Pi is de laatste versie van Debian[[2]](#footnote-2) gebruikt geweest als besturingsysteem. Met behulp van Win32DiskImager dient de SD kaart met dit besturingsysteem geinstalleerd te worden. Eenmaal de image geinstalleerd is kan men zich aanmelden op de raspberry pi met standaard credentials username = pi password= raspberry. Voor een correcte installatie verwijs ik naar de officele documentatie[[3]](#footnote-3).

Eenmaal de raspberry pi geconfigureerd is en de gebruiker zich in de home folder bevindt, kan de volgende stap ondernomen worden:

git clone:

https://github.com/VanheeTimothy/projectSmartShower/tree/master/

In de map waar Arduino.py zich bevindt zullen we eerst een map log aanmaken. Hierin komen alle errors die voorkomen bij het opstarten. Dit is optioneel maar helpt ons bij het opsporen van eventuele fouten .

Mkdir logs

belangrijk hierbij is de file arduino.py rechten krijgt, dit omdat het script moet runnen bij opstart. De u kunt de file de juiste rechten geven door volgend commando

sudo chmod +x arduino.py

vervolgens maken we een opstartscript aan op volgende locatie /bin/. Het aanmaken van het opstart script gebeurd als volgt:

sudo nano smartShower.sh

in het bestand kopieer je volgende tekst:

#!/bin/sh

sleep 15

sudo python3 /home/pi/Documents/SmartShower/arduino.py

**noot**: het pad van het .py file kan anders zijn dan deze van uw pi om de juiste directory te vinden navigeer naar Arduino.py en gebruik volgend commando pwd.

Sla smartShower.sh op door crtl+x y in te typen.

sudo chmod 755 smartShower.sh

Tot slot moeten we de crontab aanpassen

Sudo crontab -e

kopieer volgende tekst en plak deze onderaan de file:

@reboot sh /bin/smartShower.sh > /home/pi/Documents/SmartShower/logs 2>&

Indien problemen zich hiervoor doen, kan men ook kiezen voor volgende optie om het script te laten runnen at start up. Doch prefereer ik de voorgaande optie.

sudo nano /etc/profile

sudo python3 /home/pi/Documents/SmartShower/arduino.py &

**Noot**: een wanneer deze optie wordt gebruikt moet het script lichtjes aangepast worden. Hier wordt het script telkens onderbroken en krijgt een read error.

Gezien arduino.py gebruikt maakt van een packages die niet standaard met python is meegeleverd moeten we deze zelf in de terminal toevoegen. Gelukkig is pip wel standaard op de debian image geinstalleerd. Om de ontbrekende packages toe te voegen run volgende commando’s:

pip install pyserial

pip install requests

### mogelijke problemen

Indien bij de configuratie[[4]](#footnote-4) een map logs en de crontab juist configureerd is kunnen errors bij het opstarten geraadpleegd worden door volgend commando:

cd /home/pi/Documents/SmartShower/logs

cat cronlog

#### Het opstart script start niet op en print volgende error

*File "Documents/SmartShower/arduino.py", line 19, in <module>*

*line = ser.readline() # read a '\n' terminated line*

*File "/usr/lib/python2.7/dist-packages/serial/serialposix.py", line 490, in read*

*'device reports readiness to read but returned no data '*

*serial.serialutil.SerialException: device reports readiness to read but returned no data (device disconnected or multiple access on port?)*

Zorg ervoor dat de USB-kabel, tussen de Arduino en Raspberry Pi, goed en correct in de juiste poort zit. Indien het probleem nog niet is verholpen, type volgend commando in de terminal:

ls /dev/serial/by-id

Nu wordt er een id van de poort afgeprint. Kopieer het id samen met de locatie ernaar toe. Indien er geen id wordt afgeprint, controleer of de USB-kabel correct is aangesloten. Indien het probleem zich nog blijft voordoen, test voorgaande stap opnieuw met een andere USB-kabel.

We moeten nu onze arduino.py file open:

sudo nano /home/pi/Documents/SmartShower/arduino.py

Verwijder de de tekst in de serial.Serial()[[5]](#footnote-5) en plak de voorgaande gekopieerde tekst.

#### Script start niet op door ImportError:

*Traceback (most recent call last):*

*File "Documents/SmartShower/arduino.py", line 1, in <module>*

*import asyncio*

*ImportError: No module named asyncio*

**Asyncio is een python library en is compileerbaar met python3 en hoger. Zorg ervoor dat dus in je opstartscript *python3* staat en niet *python***

sudo python3 /home/pi/Documents/SmartShower/arduino.py

#### Kleurcodes bij de Raspberry Pi

Om de gebruiker een kleine indicator te geven en als gemak bij troubleshooting is er een RGB-led aan de Raspberry Pi bevestigd. Merk dus op dat dit niet noodzakelijk is om een correcte werking van de het product te hebben.

##### Succes code

Wanneer het script opstart, brandt een groene led. Indien de groene led knippert, is een gebruiker zich aan het douchen en wordt de data doorgestuurd naar de cosmosdb. Wanneer de gebruiker klaar is met douchen wordt de data van zijn sessie opgehaald om berekeningen te doen. Eenmaal de berekingen klaar zijn wordt de nieuwe data doorgestuurd naar de SQLDb. Als de status code succes 200 is zal de led kort blauw knipperen.

##### Error code

Als er een probleem is en het script het bericht niet kan decoderen zal de led Magneta kleuren, het script zal vanzelf terug opstarten.

Bij een ander error zal de led rood kleuren.Normaal hoeft u niets te ondernemen maar indien de led langer dan 5 seconden rood blijft branden, sluit de raspberry af en start deze opnieuw op.

##### Intterupt

Wanneer de gebruiker het script handmatig stop zet zal de led oranje branden. Zo weet de gerbuiker dat het script uit is maar de raspberry pi nog aan staat.

***Tip****: Voor een betere monitoring van het script, is het handig om een file met de naam error.txt aan te maken in dezelfde map waar arduino.py zich bevindt. Zo zal het script elke error met timestamp wegschrijven.*

***Indien deze stap wordt overgeslaan dienen de lijnen (106,107,117,118,128,129) in commentaar geplaatst te worden!***

# smartShower API

## languages, libraries, tools and technologies used

De API is een c# .NET cloud Microsoft.Azure Functions gebouwd in een visual studio 2017 omgeving.

Volgende assemblies, dependencies zijn gebruikt geweest voor de opbouw:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Diagnostics;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using Microsoft.Azure.Documents.Client;

using Microsoft.Azure.WebJobs;

using Microsoft.Azure.WebJobs.Extensions.Http;

using Microsoft.Azure.WebJobs.Host;

using Newtonsoft.Json;

Azure is gelinkt aan github[[6]](#footnote-6) voor synchronisatie van de code.

De entity framework is gebruikt om data op te halen en weg te schrijven naar SQL database: SmartShowerDb. Alle data die gelogt wordt door de senoren, die aan de Arduino hangen, wordt doorgestuurd naar een cosmosDb die online staat op Azure.

## Overzicht van de API

Communicatie verloopt over HTTP

Base-url: <https://smartshowerfunctions.azurewebsites.net/>

Functions.cs[[7]](#footnote-7) class neemt de volledige communciatie op zich en volgende calls zijn mogelijk:

* POST RegisterUser
  + Hier wordt een controle gedaan of het email adres al dan niet bestaat in de database. Indien dit het geval is, wordt een Forbidden status code meegeven. Wanneer het emailadres niet gekend is, wordt een nieuwe gebruiker aangemaakt in de database. Men krijgt dan status 200 OK.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/User/Reg>
* POST RegisterShower
  + Deze functie is geschreven om zelf een douche te registreren. Bij het prototype van de smartShower is de IdShower hardcode in de arduino. Dit wil zeggen dat er van deze functie momenteel geen gebruik wordt gemaakt. Toch kan deze functie in productie handig zijn. Er wordt verwacht dat men in de body een idShower en waterCost meegeeft. Standaard staat de kostprijs per liter op 0.005€.
  + RETURN 200 OK indien de functie correct uitgevoerd werd. Indien de douche al in de database zit, krijgt men een Forbidden terug.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/Shower/Reg>
* POST RegisterUserShower
  + Bij deze call wordt een gebruiker aan een douche gelinkt. In de body wordt een idUser en IdShower meegegeven.
  + RETURN indien er geen douche gevonden is, een not Found statuscode terug. Men krijgt een 200 bij succes alsook het idUser en idShower.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/UserShower/Reg>
* POST GetAvailableColors
  + Er wordt een idShower meegestuurd in de body. Door een joinquery worden alle kleuren opgevraagd die al in gebruik zijn. Vervolgens worden ze uit een list met alle kleuren verwijderd. De kleuren die overblijven zijn de kleuren die nog beschikbaar zijn.

**Noot**: voor het datatype van de kleuren is gebruikt gemaakt van integers

* + RETURN bij succes statuscode 200, en een list met kleuren. Bij een failure wordt een internalServerError weergegeven
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/GetSessions>
* POST LoginUser
  + Hier worden een email en (gehashed[[8]](#footnote-8)) wachtwoord meegeven. Er wordt een controle gedaan of deze login gegevens in de database zitten en correct zijn.
  + RETURN een status code 200 en de user zelf bij succes. De gegevens van de user worden meegestuurd zodat zijn persoonlijke instellingen kunnen worden weergeven in de applicatie. Men krijgt een Unauthorized indien de login credentials niet correct zijn.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/User/login>
* POST AddSessionToCosmosDb
  + Wordt gebruikt om de data, die de Raspberry Pi van de Arduino krijgt, door te sturen naar de CosmosDb.
  + RETURN alle data die doorgestuurd werd bij succes. Bij een faillure krijgt men een InternalServerError.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/AddSession>
* GET GetSessionFromCosmosDb
  + Aan de hand van een idSession wordt de data van deze sessie uit de cosmosdb gehaald.
  + RETURN een list met sessieData bij succes. Internalserver error bij failure.
  + [https://smartshowerfunctions.azurewebsites.net/api/SmartShower/getSession/{idSession}](https://smartshowerfunctions.azurewebsites.net/api/SmartShower/getSession/%7bidSession%7d)
* GET CalculateSession
  + Aan de hand van een idSession wordt alle data uit de cosmosdb gehaald. Hier wordt dus een GET GetSessionFromCosmosDb uitgevoerd. Met de sessiedata worden berekingen gedaan zoals gemiddelde temperatuur, waterverbruik, totale duurtijd, hoeveel geld bespaard en de ecoscore van één sessie. Nadat de berekeningen gebeurd zijn worden deze doorgestuurd naar de SQLdb in de tabel Session. Er wordt ook een timestamp toegevoegd ter controle. Aangezien een Iduser aangemaakt wordt in de applicatie zelf en niet in arduino, wordt de Iduser gelinkt aan het profielnummer die wel in de arduino bepaald wordt. Een user heeft een idUser en een kleur, de waarde van die kleur (int) is gelinkt aan het profielnummer. Zo weet men welke sessie van welke gebruiker is.  
    **Noot**: deze HTTP GET wordt aangeroepen in het pythonscript en moet asynchronious lopen, gezien de tijdsduur van deze bewerking.
  + RETURN statuscode 200 indien geslaagd. Indien er geen gebruiker kon gelinkt worden aan de het profielnummer krijgt men een NotFound error. Indien er problemen zijn aan de server kant krijgt men opnieuw een InternalServerError.
  + [https://smartshowerfunctions.azurewebsites.net/api/SmartShower/calculateSession/{idSession}](https://smartshowerfunctions.azurewebsites.net/api/SmartShower/calculateSession/%7bidSession%7d)
* POST GetSessions
  + In de body wordt een idUser meegegeven en een datalength (int). Afhankelijk van de datalength (0-2) wordt uit de database de sessie van de gebruiker weergegeven, per dag(0), per week(1) of per maand(2)
  + RETURN bij succes status code 200 en een list met sessions. Bij een failure wordt een internalServerError weergegeven.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/getAllGroupsFromUser>
* POST GetUserInfo
  + Door in de body een idUser mee te geven, krijgt men alle info van deze gebruiker terug die opgeslaan is in de tabel Users.
  + RETURN status code 200 en de userInfo. Indien geen success, wordt een InternalServerError terug gegeven.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/User/login>
* PUT UpdateUser
  + In de body wordt een object user meegegeven met daarin al zijn gegevens vervolgens worden deze gegevens van de user geupdated.
  + RETURN bij succes statuscode 200. Bij een failure krijgt men een internalServerError.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/User/Update/>
* PUT UpdateShower
  + In de body wordt een idShower meegeven en een float waterCost. Vervolgens worden de gegevens geupdate in de database.
  + RETURN status code 200 bij succes, internalServerError bij een fout
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/User/Update/>
* POST MakeGroup
  + Hier wordt een idGroup en idUser meegeven. Pending van de gebruiker staat automatisch op 0. Dit wil zeggen dat wanneer een gebruiker een groep aanmaakt, hij zichzelf niet hoeft te accepeteren in zijn eigen groep.
  + RETURN statuscode OK, true bij succes. Bij failure krijgt de gebruiker een internalServerError.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/Group/new/>
* DELETE DeleteGroup
  + In de body wordt een IdGroup meegegeven. Vervolgens wordt een query uitgevoerd die de groep met het desbetreffende id verwijderd.
  + RETURN statuscode OK, true bij succes. Bij failure krijgt de gebruiker een internalServerError.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/Group/delete/>
* DELETE DeleteUserFromGroup

In de body wordt een idGroup en Iduser meegeven. Vervolgens zal de rij uit de database verwijderd worden waar idGroup en IdUser gelinkt zijn.   
**Noot**: deleteUserFromGroup en DeclineInvite hebben allebei dezelfde functie. Vandaar er enkel DeleteUserFromGroup aangemaakt is.

* + RETURN statuscode OK, true bij succes. Bij failure krijgt de gebruiker een internalServerError.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/Group/user/delete>
* POST SendGroupInvite
  + Het verzenden van GroupInvites gebeurd via het email adres van een gebruiker. Van deze gebruiker worden gegevens opgehaald zoals, idUser, naam, kleur en foto. Indien een gebruiker is gevonden in de database wordt de idUser wordt gelinkt aan de idGroup.
  + RETURN bij succes statuscode 200 en de idUser, naam, kleur en foto meegestuurd. Indien er geen gebruiker gevonden werd wordt een statuscode NotFound meegestuurd. Bij een andere failure krijgt men een internalServerError.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/Group/invite>
* PUT AcceptInvite
  + Zet de pending van de meegegeven (id)User op 0. Dit betekent dat de gebruiker een uitnodiging van een bepaalde groep heeft goedgekeurd.
  + RETURN true indien de HTTP verb correct is uitgevoerd. InternalServerError indien de operatie niet geslaagd is.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/Group/invite/accept>
* POST GetAllGroupsFromUser
  + Nadat een idUser is meegeven in de body worden alle idGroups (met groepsnaam en foto) waar de gebruiker in zit teruggegeven.
  + RETURN bij succes een list met Usergroups. Deze bevat objecten met properties: IdGroup, naam, groepsfoto en IdUser, . Bij een failure wordt een internalServerError meegeven.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/getAllGroupsFromUser>
* POST GetAllInvitesFromUser
  + Nadat een idUser is meegeven in de body worden ook hier alle groupdata terug geven waar de user nog geen invite van heeft geaccepteerd.
  + RETURN bij succes een list met Usergroups. Deze bevat objecten met properties: IdGroup, naam, groepsfoto en IdUser, . Bij een failure wordt een internalServerError meegeven.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/getAllGroupsFromUser>
* POST GetAllFriendsFromUser
  + In de body een idUser meegegeven en een boolean withSessionData. In de database wordt gekeken in welke groepen deze idUser terug te vinden is. Vervolgens wordt in deze groepen gekeken welke andere gebruikers erin zitten. Deze personen worden beschouwd als vrienden van de gebruiker. Indien withSessionData = true dan word ook de sessiedata van de gebruiker meegegeven.
  + RETURN bij succes statuscode 200 en een list met users (& hun sessiedata) die een gemeenschappelijke groep delen. Bij failure krijgt de gebruiker een internalServerError.
  + <https://smartshowerfunctions.azurewebsites.net/api/SmartShower/getAllGroupsFromUser>

## klassen

bij de API horen volgende klassen:

* Session.cs
* SessionCosmosDb
* GroupSessions.cs
* Shower.cs
* User.cs
* UserGroup.cs
* UserShower.cs

De klasses zijn terug te vinden in het project SmartShowerFunctions in de map Model.

## Configuratie

Om de azure functions lokaal te kunnen draaien moeten de rode variabelen correct worden toegevoegd in de local.settings.json:

{

"IsEncrypted": false,

"Values": {

"AzureWebJobsStorage": "",

"AzureWebJobsDashboard": "",

"ConnectionString": "",

"COSMOSHOST": "",

"COSMOSKEY": "",

"COSMOSDATABASE": "",

"COSMOSCOLLECTIONID": ""

}

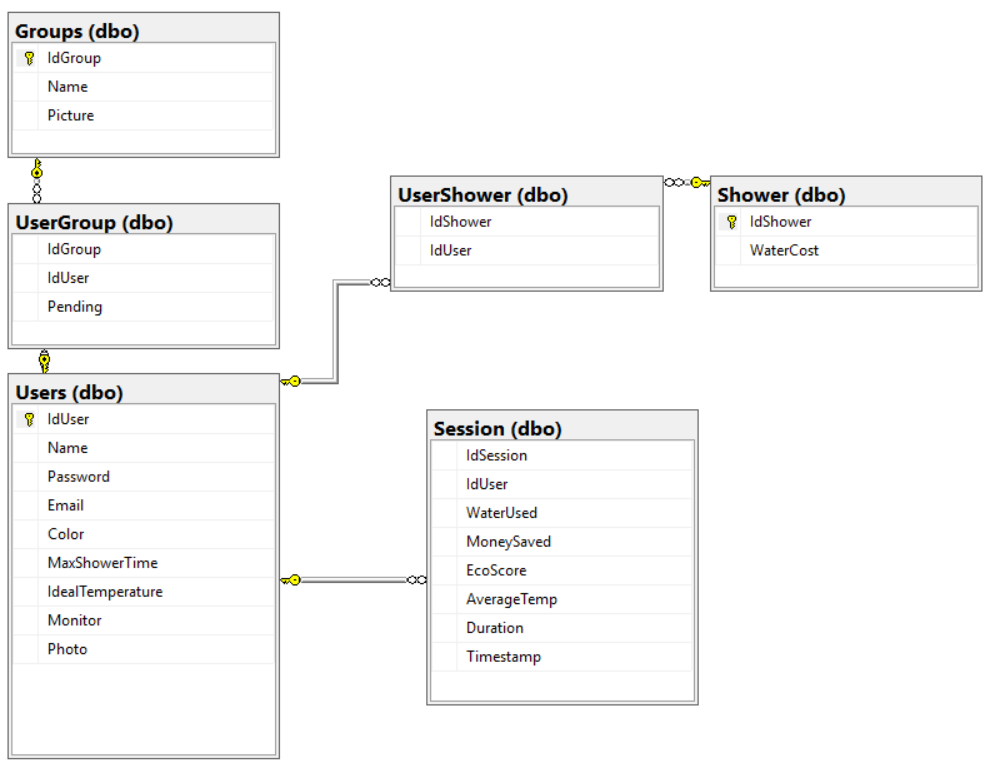
}

Waneer men alles online wilt testen mag men niet vergeten de connectionString toe te voegen aan de azure function. Ook aan de cosmosDb moeten zoals de host, key, databaseName en cosmoscollectionId worden gecontroleerd zodat alle waarden juist zijn.

# Databases

## MSSQL: SmartShowerDb

### ERD-schema



ERD-schema SmartShower

### Tabellen

#### Users

|  |  |  |
| --- | --- | --- |
| ColumnName | Datatype | Allows null |
| idUser | Unqiueidentifier | No |
| Name | Varchar(150) | No |
| Password | Varchar(150) | No |
| Email | Varchar(150) | No |
| Color | Int | No |
| MaxShowerTime | Int | No |
| IdealTemperature | Float | No |
| Monitor | tinyInt | No |
| Photo | Image | yes |

**Noot**: Monitor is een tiny Int en fungeert als een Boolean. Het is om bij te houden of een gebruiker al dan niet wilt dat zijn gegevens van een sessie mogen worden gebruikt om een betere sessie voor te stellen waarbij de gebruiker eventueel meer water en geld kan uitsparen. Bij het prototype smartShower ontbreekt het learning process waarbij de smartShower zelf betere doucheschema’s voorstelt.

#### Session

|  |  |  |
| --- | --- | --- |
| ColumnName | Datatype | Allows null |
| IdSession | Uniqueidentifier | No |
| IdUser | Uniqueidentifier | No |
| WaterUsed | Float | No |
| MoneySaved | Float | No |
| Ecoscore | Float | No |
| AverageTemp | Float | No |
| Duration | Time(2) | No |
| Timestamp | Datetime | No |

**Noot**: deze data is een bereking van de data die opgehaald wordt uit de cosmosdb

#### Groups

|  |  |  |
| --- | --- | --- |
| ColumnName | Datatype | Allows null |
| IdGroup | Uniqueidentifier | No |
| Name | Varchar(150) | No |
| Picture | Image | yes |

#### UserGroup

|  |  |  |
| --- | --- | --- |
| ColumnName | Datatype | Allows null |
| IdGroup | Uniqueidentifier | No |
| IdUser | Uniqueidentifier | No |
| Pending | tinyInt | No |

**Noot**: Pending is een boolean die fungeert als controle of een gebruiker een groepsinvite heeft geaccepteerd.

#### Shower

|  |  |  |
| --- | --- | --- |
| ColumnName | Datatype | Allows null |
| IdShower | Uniqueidentifier | No |
| WaterCost | Float | No |

#### UserShower

|  |  |  |
| --- | --- | --- |
| ColumnName | Datatype | Allows null |
| IdShower | Uniqueidentifier | No |
| IdUser | Uniqueidentifier | No |

### Overzicht gebruikte queries

#### RegisterUser

select Email FROM Users Where Email = @mail

INSERT INTO Users VALUES(@Guid, @Name, @Password, @Email, @Color, @MaxShowerTime, @IdealTemperature, @Monitor, @Photo)

#### RegisterShower

SELECT \* FROM Shower WHERE IdShower = @ShowerId;

INSERT INTO Shower VALUES(@IdShower, @WaterCost)

#### RegisterUserShower

SELECT \* FROM Shower WHERE IdShower = @ShowerId;

INSERT INTO UserShower VALUES(@IdShower,@IdUser)

#### LoginUser

SELECT \* FROM Users where Email like @Email and Password = @password;

#### GetUserInfo

SELECT Name, Email, Color, MaxShowerTime, IdealTemperature, Monitor, Photo FROM Users WHERE IdUser Like @IdUser;

#### UpdateUser

UPDATE Users SET Name = @Name, Password = @Password, Email = @Email, Color = @Color, MaxShowerTime = @MaxShowerTime, IdealTemperature = @IdealTemperature, Monitor = @Monitor, Photo = @Photo WHERE IdUser = @IdUser;

#### UpdateShower

UPDATE Shower SET WaterCost = @Watercost WHERE IdShower = @showerId;

#### MakeGroup

INSERT INTO UserGroup VALUES(@IdGroup, @IdUser, 0);

#### DeleteGroup

DELETE FROM UserGroup WHERE IdGroup = @IdGroup;

#### DeleteUserFromGroup

DELETE FROM UserGroup WHERE IdGroup = @IdGroup AND IdUser = @IdUser ;

#### SendGroupInvite

SELECT IdUser, Name, Color, Photo FROM Users WHERE Email = @Email;

INSERT INTO UserGroup VALUES(@IdGroup, @IdUser, 1);

#### AcceptInvite

UPDATE UserGroup SET Pending = 0 WHERE IdGroup = @IdGroup AND IdUser = @IdUser;

#### GetAllGroupsFromUser

#### SELECT Groups.IdGroup, Name, Picture FROM dbo.Groups INNER JOIN UserGroup ON Groups.IdGroup = dbo.UserGroup.IdGroup WHERE IdUser = @IdUser AND dbo.UserGroup.Pending = 0;

#### GetAllInvitesFromUser

#### SELECT Groups.IdGroup, Name, Photo FROM dbo.Groups INNER JOIN UserGroup ON Groups.IdGroup = dbo.UserGroup.IdGroup WHERE IdUser = @IdUser AND dbo.UserGroup.Pending = 1;

#### GetAllFriendsFromUser

#### SELECT DISTINCT UserIdTable.[IdUser], Users.[Name], Users.[Photo], Sessions.WaterUsed, Sessions.MoneySaved, Sessions.EcoScore, Sessions.AverageTemp, Sessions.Duration, Sessions.Timestamp FROM[dbo].[UserGroup] AS UserIdTable INNER JOIN[dbo].[UserGroup] AS GroupIdTable ON GroupIdTable.[IdGroup] = UserIdTable.[IdGroup] and GroupIdTable.[IdUser] = '06B247E1-EEA4-EF62-1111-A7838E4E0DBF' INNER JOIN [dbo].[Users] as Users ON Users.[IdUser] = UserIdTable.[IdUser] INNER JOIN dbo.Session AS Sessions ON Sessions.IdUser = Users.IdUser;

#### SELECT DISTINCT UserIdTable.[IdUser], Users.[Name], Users.[Photo] FROM[dbo].[UserGroup] AS UserIdTable INNER JOIN[dbo].[UserGroup] AS GroupIdTable ON GroupIdTable.[IdGroup] = UserIdTable.[IdGroup] and GroupIdTable.[IdUser] = @IdUser INNER JOIN [dbo].[Users] as Users ON Users.[IdUser] = UserIdTable.[IdUser];

#### GetSessions

#### SELECT IdSession, IdUser, WaterUsed, MoneySaved, EcoScore, AverageTemp, Duration, Timestamp FROM Session WHERE IdUser = @IdUser AND Timestamp >= DATEADD(DAY, 0, DATEDIFF(DAY, 0, CURRENT\_TIMESTAMP)) AND Timestamp < DATEADD(DAY, 1, DATEDIFF(DAY, 0, CURRENT\_TIMESTAMP));

#### SELECT IdSession, IdUser, WaterUsed, MoneySaved, EcoScore, AverageTemp, Duration, Timestamp FROM Session WHERE IdUser = @IdUser AND Timestamp >= DATEADD(DAY, 0, DATEDIFF(DAY, -7, CURRENT\_TIMESTAMP)) AND Timestamp < DATEADD(DAY, 1, DATEDIFF(DAY, 0, CURRENT\_TIMESTAMP));

#### SELECT IdSession, IdUser, WaterUsed, MoneySaved, EcoScore, AverageTemp, Duration, Timestamp FROM Session WHERE IdUser = @IdUser AND Timestamp >= DATEADD(DAY, 0, DATEDIFF(DAY, -30, CURRENT\_TIMESTAMP)) AND Timestamp < DATEADD(DAY, 1, DATEDIFF(DAY, 0, CURRENT\_TIMESTAMP));

#### CalculateSession

#### select Users.IdUser from Users INNER JOIN UserShower ON Users.IdUser = UserShower.IdUser WHERE IdShower = @IdShower AND Users.Color = @Color;

#### INSERT INTO Session VALUES(@IdSession, @IdUser, @WaterUsed, @MoneySaved, @EcoScore, @AverageTemp, @Duration, @Timestamp);

#### GetAvailableColors

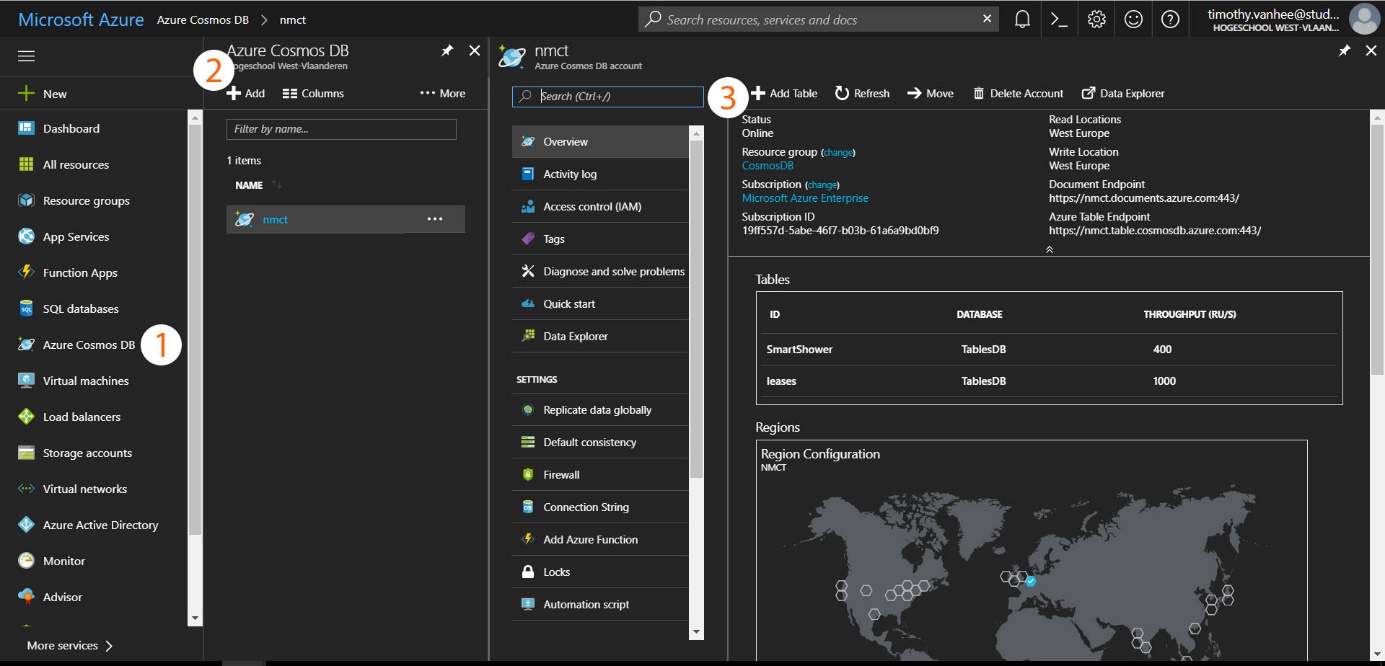
#### select color from Users inner join UserShower on Users.IdUser = UserShower.IdUser where UserShower.IdShower = @IdShower;

## CosmosDb smartShower

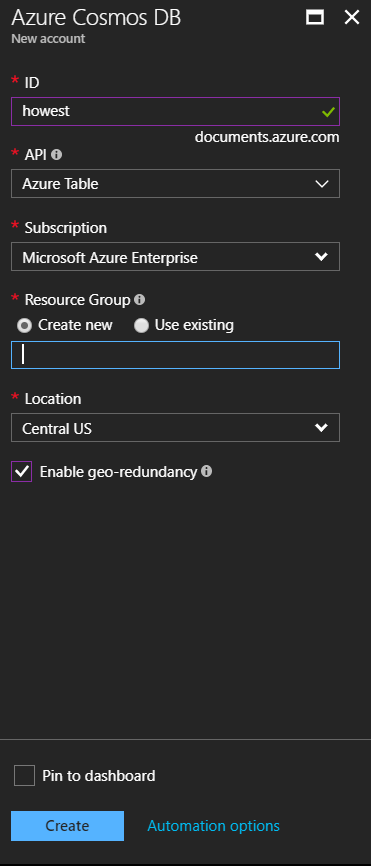
*Wanneer een douchesessie van start gaat, wordt elke seconde data verstuurd en opgeslaan. Hierdoor wordt een grote hoeveel aan data gecreerd. Dit wordt dan nog vermenigdvuldigd met het aantal gebruikers. Uitgaande van deze gedachte is besloten om data niet op te slaan in een SQLdb maar de alomwelbekende NoSql Cosmosdb.*

### Aanmaak van de database

Voor het project Smart Shower is een NoSql database aangemaakt op Azure. Via deze [link](https://docs.microsoft.com/en-us/azure/cosmos-db/introduction) kunt u gratis de cosmosdb uitproberen.



Dashboard Azure Function

Eenmaal uw account is aangemaakt klikt u op Azure CosmosDB (1). Vervolgens op + Add(2) dan krijgt u volgend scherm te zien:

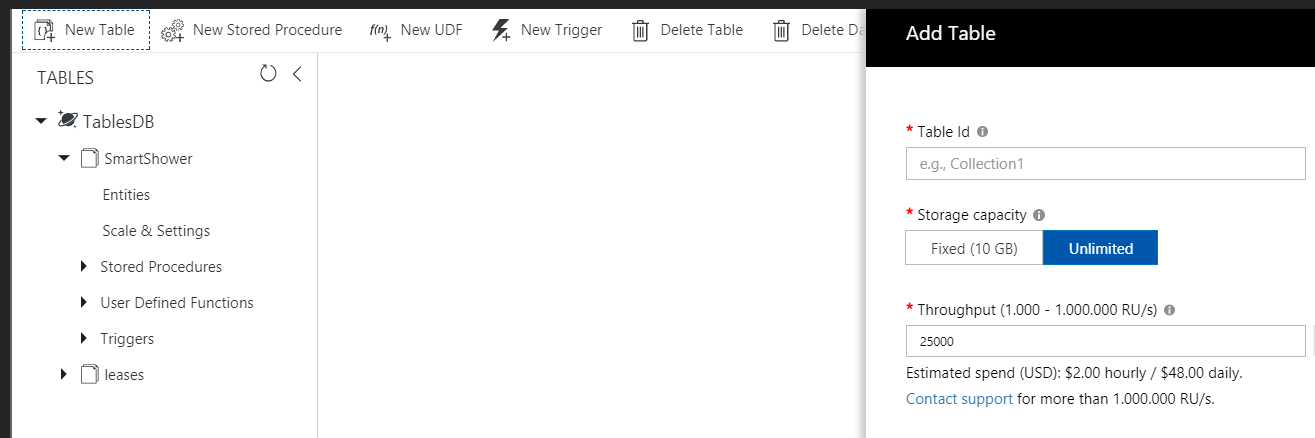
Add database

Vul een ID in, kzie bij API voor Azure Table, kies een subscription, een Resource Group en locatie naar keuze. Tot slot klikt u op de Azure blauwe create knop.



Create cosmosdb button

Eenmaal de CosmosDb is aangemaakt klikt u op + add Table. U bevindt zich nu in de data explorer.



Data Explorer

Kies een geldig table id. Voor het prototype is gekozen voor de laagste Troughput 1000 en fixed storage capacity. Merkop eenmaal in productie zullen deze waarden anders zijn naargelang de hoeveelheid data.

**Noot**: voor de aanmaak van deze database is ook een handige [applicatie](https://azure.microsoft.com/en-us/features/storage-explorer/).

### Overzicht gebruikte queries

#### AddSessionToCosmosDb

IQueryable<SessionCosmosDb> logs = client.CreateDocumentQuery<SessionCosmosDb>($"/dbs/{COSMOSDATABASE}/colls/{COSMOSCOLLECTIONID}", new FeedOptions { EnableCrossPartitionQuery = true, MaxDegreeOfParallelism = 10, MaxBufferedItemCount = 100 }).Where(p => p.IdSession == new Guid(idSession));

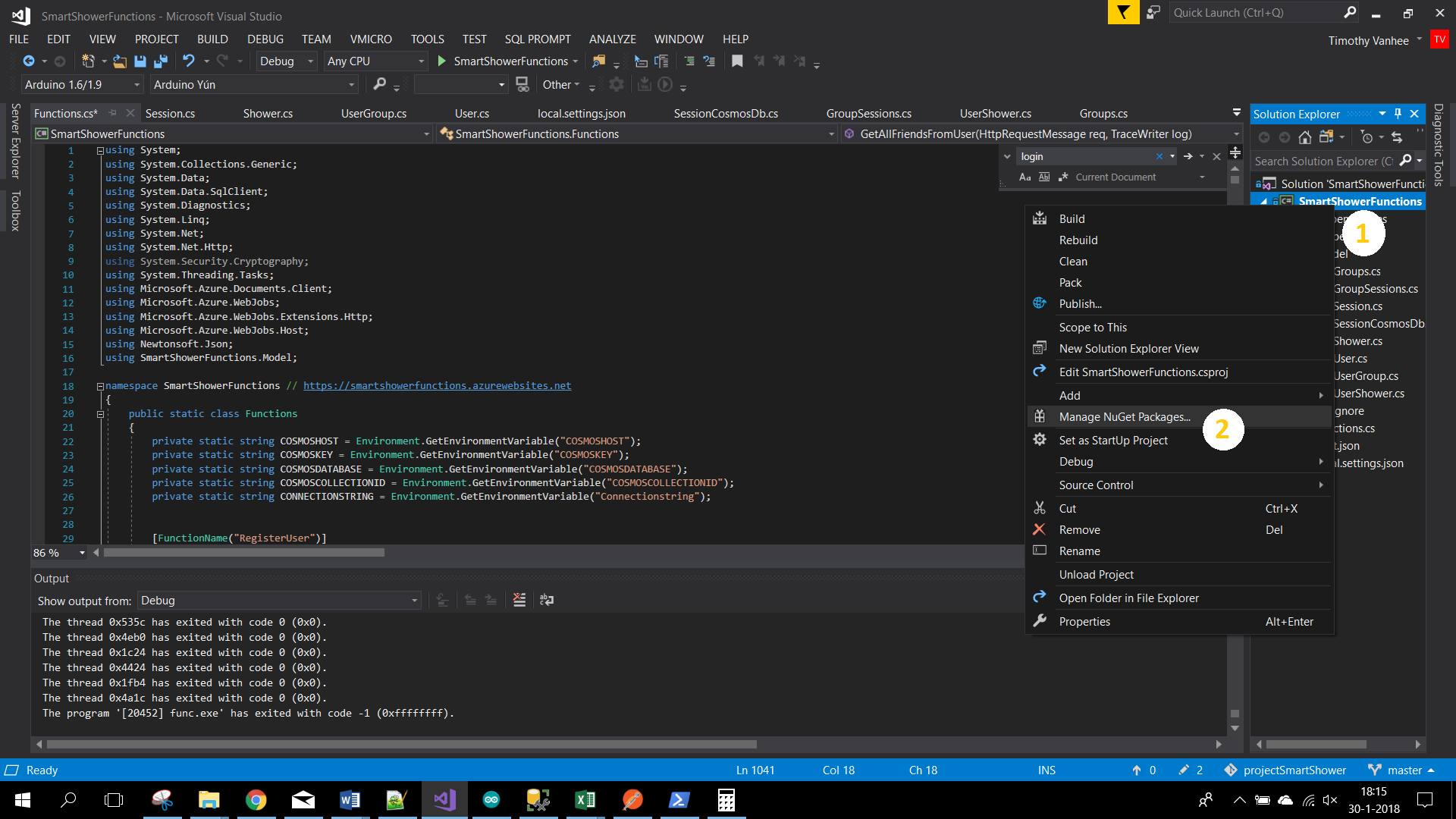
# SmartShowerApp Cross-Platform (Xamarin.Forms)

## languages, libraries, tools and technologies used

De software is een c# Cross-Platform App (Xamarin.forms) gebouwd in een visual studio 2017 omgeving. (.NET Framework 4.6.1)

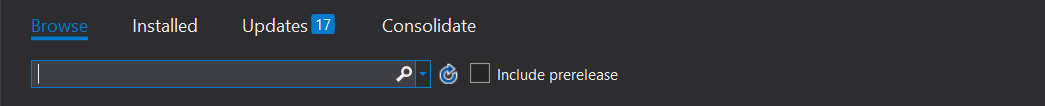
**Noot**: voor een correcte installatie van de nodige software verwijs ik naar de officiele [documentatie](https://developer.xamarin.com/guides/cross-platform/getting_started/installation/windows/).

## Extra nuget packages

Bij de ontwikkeling van de applicatie Smart Shower zijn extra packages toegevoegd. Om dit te kunnen doen gaan we naar het PCL project en klikken we rechter-muisklik(1). Vervolgens kiezen we voor *Manage NuGet Packages(2)*.

Ga naar NuGet-manager

Eenmaal aangekomen in de NuGet Manager kunnen we in de search-balk de naam van het gewenste package invullen.

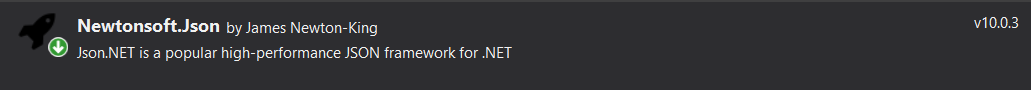


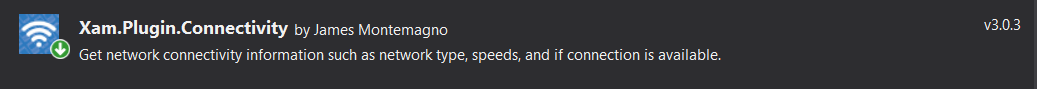
Search-balk NuGet-manager

## Lijst extra geinstalleerde packages

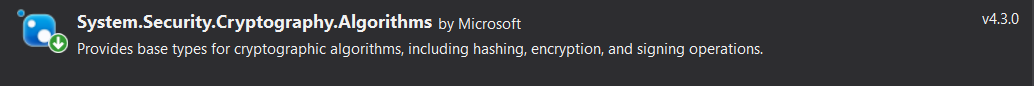
Hieronder vindt u een lijst alle extra geinstalleerde NuGet-packages en hun versie.











## Structuur PCL project

### Assets

In de map Assets bevinden zich de gebruikte .png files(8).

### Model

#### AppElements.cs

bevat 4 subklassen, USP, Details, Ranking, Rank

* USP
  + (string) Name
  + (ImageSource) ImageSrc
* Details
  + (string) Name
  + (string) Value
  + (ImageSource) ImageSrc
* Ranking
  + (List<Rank>)Ranks
* Rank
  + (string) RankNo
  + (Guid) IdUser
  + (string) Name
  + (string) Photo
  + (string) Value

#### User.cs.

Deze klasse bevat alle attributen die in de database opgeslaan worden in de juiste kolom onder de correcte tabel.

* User
  + (Guid) IdUser
  + (string) Name
  + (string) Password
  + (string) email
  + (int) color
  + (int) MaxShowerTime
  + (float) IdealTemperature
  + (bool) monitor
  + (string) photo
* Shower
  + (Guid) IdShower
  + (Guid) IdUser
* SessionInfo
  + (Guid) IdUser
  + (int) Datalength
  + (bool) WithSessionData
* Session
  + (Guid) IdGroup
  + (Guid) IdUser
  + (string) Name
  + (string) Photo
  + (float) WaterUsed
  + (float) MoneySaved
  + (float) EcosScore
  + (float) AverageTemp
  + (TimeSpan) Duration
* Group
  + (Guid) IdGroup
  + (string) Name
  + (string) Photo
  + (Guid) IdMaker
  + (string) EmailInvitee

#### Manager.cs.

Van hieruit wordt de API aangesproken en beheert.

|  |  |
| --- | --- |
| LoginUserAsync  RegisterUserAsync  UpdateUserAsync  UpdateShowerAsync  GetUserInfoAsync  RegisterShowerUserAsync  RegisterShowerUserAsync  GetSessionsAsync | GetAllGroupsAsync  GetAllFriendsAsync  GetSessionsShowerAsync  GetAvailableColorsAsync  RegisterGroupAsync  SendInviteGroupAsync  CreateKey (dit is een functie om de wachtwoorden te hashen) |

Voor een dieper inzicht over de implementatie van de API, in de applicatie, verwijs ik naar de klasse Manager.cs die zich in de map model bevindt onder het PCL project.

Alle bovenstaande functies zijn private. Ze worden aangeroepen in een publieke functie en die de waarde van de private functies returnt. Op die manier wordt de integriteit van de code behouden.

# Conclusies, opmerkingen en suggesties

## RTC module Arduino

Momenteel wordt de duurtijd van een douchesessie berekent op het aantal records die worden doorgestuurd. Elke seconde wordt data van de arduino verzonden naar de raspberry pi dwz 30 records = 30 seconden douchen. Eerst werd een timestamp meegegeven vanuit de raspberry pi voordat deze de data doorstuurde naar de cosmosdb. Dit zorgde ervoor dat één douchebeurt gelijk stond aan de tijd die de raspberry pi nodig had om alle data te versturen naar de cosmosdb.

Een accuratere oplossing zou zijn om een [RTC](https://www.aliexpress.com/item/Free-shipping-20pcs-lot-The-Tiny-RTC-I2C-modules-24C32-memory-DS1307-clock-RTC-module-for/1876368739.html?spm=2114.search0104.3.1.42c9089b7Kicfe&ws_ab_test=searchweb0_0,searchweb201602_1_10065_10068_10344_10342_10343_10340_10341_10084_10083_10618_10304_10615_10307_10302_10313_10059_10534_100031_10103_441_10624_442_10623_10622_10621_10620_10142,searchweb201603_2,ppcSwitch_7&algo_expid=8c976fe0-6b3b-4526-9a81-fb364d23e68c-0&algo_pvid=8c976fe0-6b3b-4526-9a81-fb364d23e68c&priceBeautifyAB=1) module aan de arduino te sluiten. Dit kan via AliExpress voor 0.55€.

## RFID

De douche heeft momenteel een GUID deze is nu hardcoded in de raspberry pi. Het zou een betere oplossing zijn om een [RFID](https://www.aliexpress.com/item/PN532-NFC-RFID-Module-V3-Kits-Reader-Writer/32452824672.html?spm=2114.search0104.3.1.7b303f4aFSMWFa&ws_ab_test=searchweb0_0,searchweb201602_1_10065_10068_10344_10342_10343_10340_10341_10084_10083_10618_10304_10615_10307_10302_10313_10059_10534_100031_10103_441_10624_442_10623_10622_10621_10620_10142,searchweb201603_2,ppcSwitch_7&algo_expid=5c3517f5-b923-4dab-9c94-78608a7b6b72-0&algo_pvid=5c3517f5-b923-4dab-9c94-78608a7b6b72&priceBeautifyAB=1) scanner te gebruiken.

1. Dit script is meegeleverd met dit document en bevindt zich in de serial\_test2 map [↑](#footnote-ref-1)
2. https://www.raspberrypi.org/downloads/raspbian/

   [↑](#footnote-ref-2)
3. <https://www.raspberrypi.org/documentation/configuration/raspi-config.md> [↑](#footnote-ref-3)
4. Zie Setup configuratie Raspberry Pi [↑](#footnote-ref-4)
5. Code line: 9 Arduino.py [↑](#footnote-ref-5)
6. https://github.com/VanheeTimothy/projectSmartShower.git [↑](#footnote-ref-6)
7. Voor gebruikte queries zie: hoofdstuk Database SmartShower *overzicht gebruikte queries* [↑](#footnote-ref-7)
8. Protocol SHA256 [↑](#footnote-ref-8)