Basis IoT (Internet of Things)

Realiseren

3

Een API opzetten

Algemene informatie

Onderwerp	Een API opzetten
Leerdoel(en)	1. De student kan een Azure Web App aanmaken 2. De student kan code via Git naar deze Web App pushen 3. De student kan data via een API publiek beschikbaar maken 4. De student kan data uit de DB via een JSON response versturen
Vereiste voorkennis	1. 1 De student kan via Python een MSSQL benaderen 2. De student kan een Azure gedeelde database opzetten 3. De student kan de data van een sensoren in een database plaatsen
Kwalificatiedossier	☐ B1-K1-W1: Plant werkzaamheden en bewaakt de voortgang ☐ B1-K1-W2: Ontwerpt software ☒ B1-K1-W3: Realiseert (onderdelen van) software ☐ B1-K1-W4: Test software ☐ B1-K1-W5: Doet verbetervoorstellen voor de software ☐ B1-K2-W1: Voert overleg ☐ B1-K2-W2: Presenteert het opgeleverde werk ☐ B1-K2-W3: Reflecteert op het werk

Inhoudsopgave

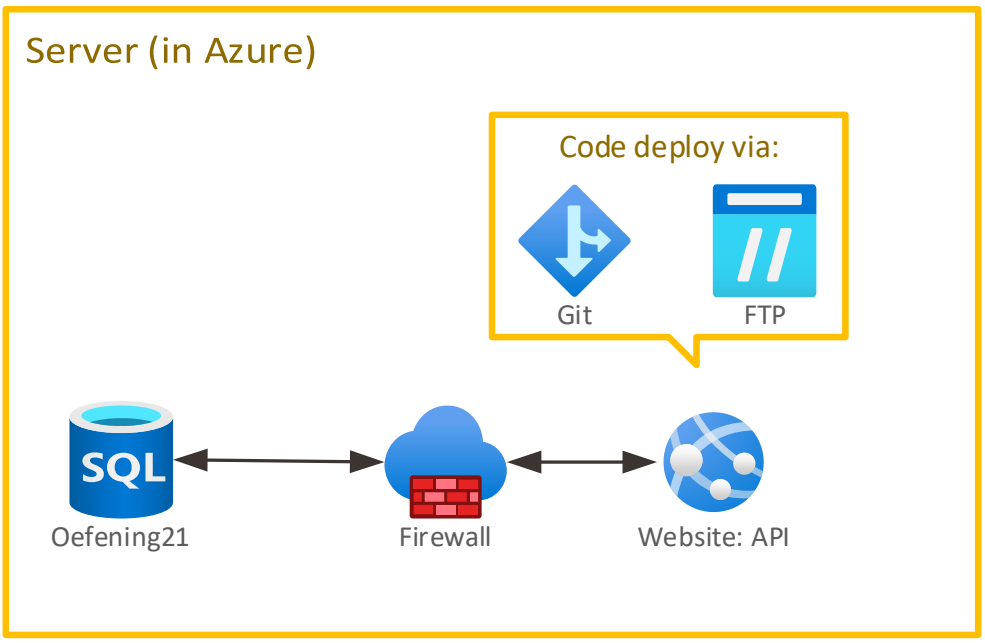
Introductie

Je hebt in het vorige hoofdstuk geleerd hoe een Raspberry PI werkt en hoe je via dit apparaat enkele sensoren kunt uitlezen. De data van deze sensoren komen via de programmeertaal Python (eigenlijk is Python een scripttaal) binnen.

Je hebt ook geleerd hoe je deze data in een (online) database kunt laten opslaan, zodat deze data bijvoorbeeld weer beschikbaar is in een C# applicatie.

Echter heb je één probleem. En dat is het feit dat deze data niet publiek beschikbaar is (omdat je niet iedereen zomaar je inloggegevens van je database wilt geven). En daarnaast heb je het probleem dat apps (bijvoorbeeld Xamarin) niet zomaar een MSSQL-database kunnen uitlezen.

Daarom ga je in dit hoofdstuk leren hoe je zelf een API kunt bouwen, waarmee mensen/gebruikers (lees: andere software developers) jouw data kunnen benaderen. Je gaat dus leren om zelf een API op te zetten.



Inhoud

1. Wat gaan we maken?

In de vorige reader hebben we geleerd hoe je via een RaspberryPI de data van een bepaalde sensor (temperatuur, beweging, o.i.d.) kunt versturen naar een Azure Database. Hieronder zie je een voorbeeld van de SensorData table:

Results Messages

	sensorDevice	sensorValue	createDateTime
1	thermometer1	19,8	2022-12-12 09:45:00.000
2	thermometer1	20,1	2022-12-12 09:47:00.000
3	thermometer1	20,3	2022-12-12 09:51:00.000

Deze data staat dus opgeslagen in de Azure omgeving binnen een MSSQL-database. Zodra we deze data publiek willen maken (voor andere applicaties/software) levert dit een probleem op betreffende de inloggegevens van de database. We willen namelijk de username en password van de MSSQL-server en database niet zomaar aan een ander geven.
Daarom dat we een webserver gaan inrichten die als Server de JSON string zal gaan publiceren aan het publiek.

Voorbeeld van de JSON string van de SensorData-tabel:


```
[{"sensorDevice":"thermometer1","sensorValue":"19,8","createDateTime":{"date":"2022-12-12
09:45:00.000000","timezone_type":3,"timezone":"UTC"}},{"sensorDevice":"thermometer1","sensorValue":"20,1","createDateTime":
{"date":"2022-12-12 09:47:00.000000","timezone_type":3,"timezone":"UTC"}},
{"sensorDevice":"thermometer1","sensorValue":"20,3","createDateTime":{"date":"2022-12-12
09:51:00.000000","timezone_type":3,"timezone":"UTC"}}]
```


Bovenstaande JSON-string is voor een mens moeilijk leesbaar (maar niet voor de computer), echter als we hem door een JSON viewer halen zien we onderstaande:


```
▼ [{sensorDevice: "thermometer1", sensorValue: "19,8",…},…]
  ▼ 0: {sensorDevice: "thermometer1", sensorValue: "19,8",…}
    ▶ createDateTime: {date: "2022-12-12 09:45:00.000000", timezone_type: 3, timezone: "UTC"}
      sensorDevice: "thermometer1"
      sensorValue: "19,8"
  ▼ 1: {sensorDevice: "thermometer1", sensorValue: "20,1",…}
    ▶ createDateTime: {date: "2022-12-12 09:47:00.000000", timezone_type: 3, timezone: "UTC"}
      sensorDevice: "thermometer1"
      sensorValue: "20,1"
  ▼ 2: {sensorDevice: "thermometer1", sensorValue: "20,3",…}
    ▶ createDateTime: {date: "2022-12-12 09:51:00.000000", timezone_type: 3, timezone: "UTC"}
      sensorDevice: "thermometer1"
      sensorValue: "20,3"
```


Een API is dus de vertaalslag, waarin we gegevens uit een MSSQL-database aan het publiek kunnen uitdelen via een zogeheten JSON-string.

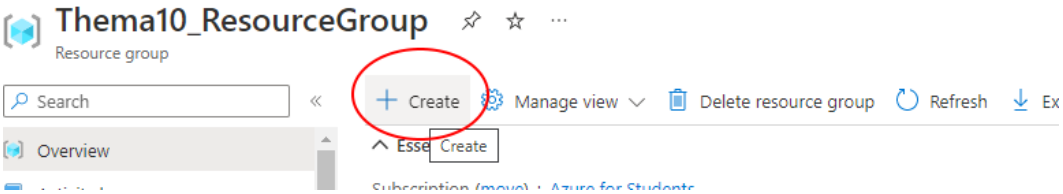
2. Voorbereiding: Een PHP webserver opzetten via Azure

Voeg aan de eerder gemaakte resourcegroup (Thema10_ResourceGroup) een zogehete WebApp toe. Binnen deze Web App gaan we een PHP-script schrijven die als API gaat fungeren.

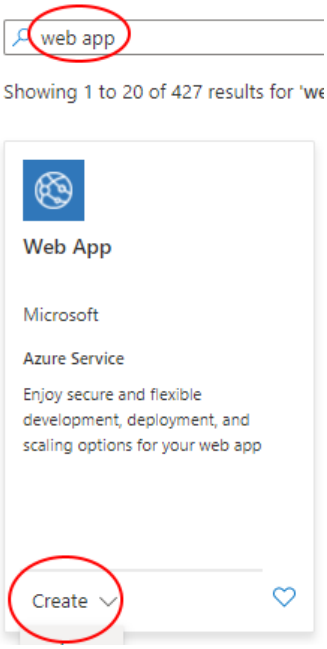
Onderstaande acties volg je laptop / computer (dus niet via de Raspberry PI).

2.1 Aanmaken Web App in Azure

➔ Ga naar de gevraagde resourcegroup en klik op Create.

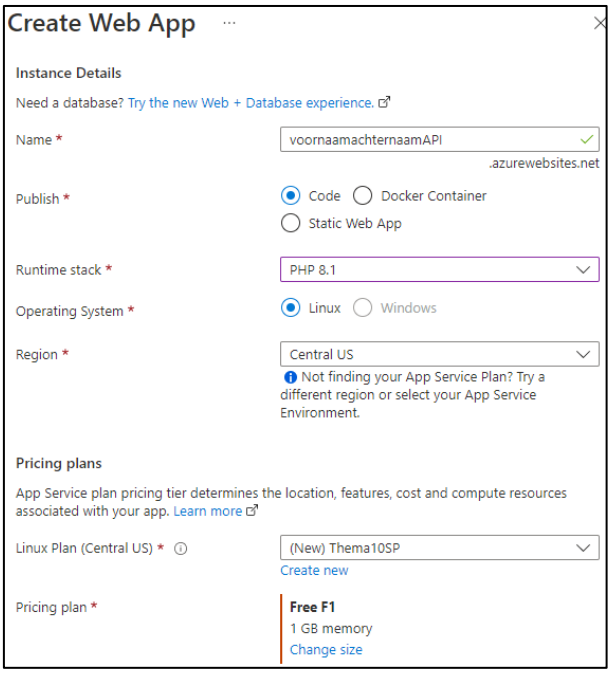


➔ Zoek naar "Web App" en klik op Create.

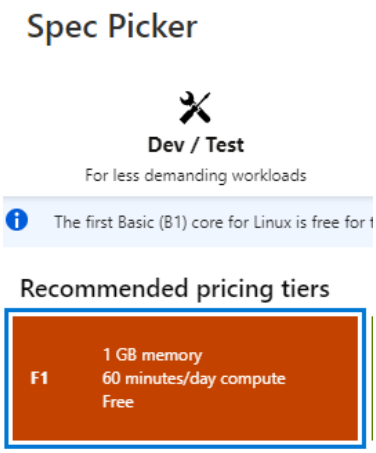


Geef deze Web App de volgende settings:

Resource Group:	Thema10_ResourceGroup
Name:	*jouwvoornaamachternaam*API
Publish:	Code
Runtime stack:	PHP 8.? ← Kies de hoogste PHP versie
Region:	Central US ←Dit omdat hier een gratis pricing plan aanwezig is.
Linux Plan:	Create new → Noem deze Thema10SP
Pricing Plan:	Free - F1 (Zie afbeelding hieronder)



➔ Opgelet: Zet het pricing plan op "Free". Doe je dat niet, dan gaat deze Web App wederom van je krediet af. Vergeet niet op Apply te drukken 😄



➔ Create nu de Web App. Dit kan enkele minuten duren

➔ Ga nu naar het dashboard van jouw aangemaakte Web App.

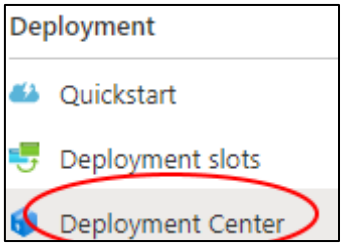
2.2 Deployment via Git naar de Web App mogelijk maken

Op dit moment kunnen we alleen code naar deze Web App sturen via FTP. Dit is nogal een oud protocol en wordt nauwelijks meer serieus gebruikt in de "echte wereld". Daarom dat wij een verbinding via een Git repository gaan opzetten, zodat wij gemakkelijk code kunnen commit en pushen naar Azure.

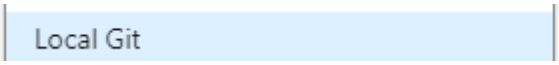
Protip:
Kom je er echt niet uit om via Git code naar deze Web App te sturen, dan zou je nog via FTP (installeer FileZilla → https://filezilla-project.org) kunnen proberen.

We gaan dus Git als source instellen. Het proces van je code online plaatsen noemen we deployen.

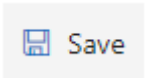
➔ Ga naar het Deployment Center van de Web App:



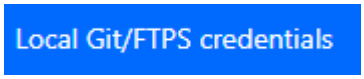
➔ Kies als Source een Local Git:



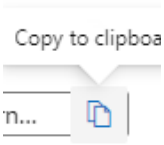
➔ Druk op Save:



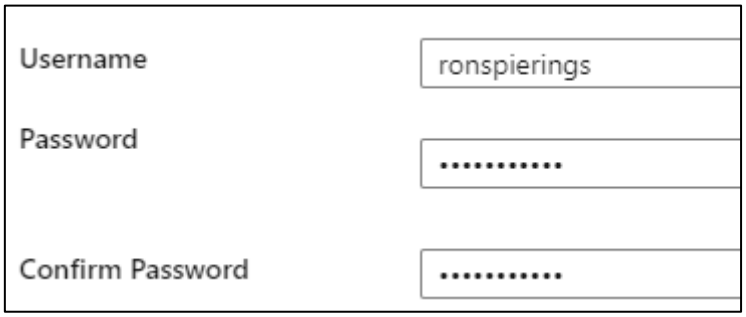
➔ Ga naar Local Git settings:



➔ Je ziet hier de Git Clone Uri. Deze hebben we later nodig, dus kopieer deze vast:



➔ Geef onder de User Scope een username en password op. Noteer deze username en password, want je hebt ze later weer nodig. Wellicht moet je de username en password ook delen met een andere student of docent, dus gebruik geen password wat je vaker gebruikt.



➔ Druk weer op Save. Dubbelcheck of je username ook daadwerkelijk nu in het veld komt te staan.

Gefeliciteerd! Je hebt zojuist je Web App open gesteld om via Git code te deployen (via commit en push). In het volgende hoofdstuk gaan we de PHP-code schrijven (via Visual Studio Code) om jouw API aan de gang te slingeren!

2.3 Git commit en push via Visual Studio

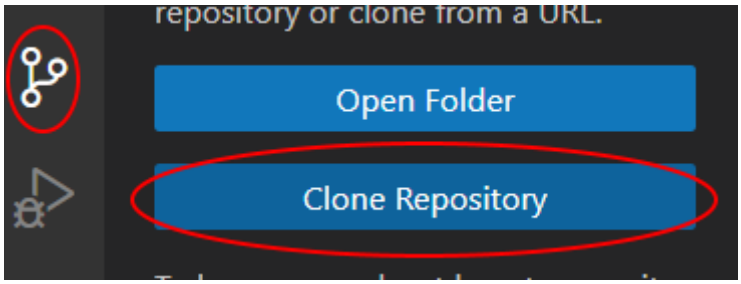
Laten we de gemaakte Git repository inladen. Daarna gaan we de PHP-code schrijven en deze via commit en push in Visual Studio Code, om de code te deployen naar Azure.

➔ Download Visual Studio Code (https://code.visualstudio.com/). Open deze applicatie.

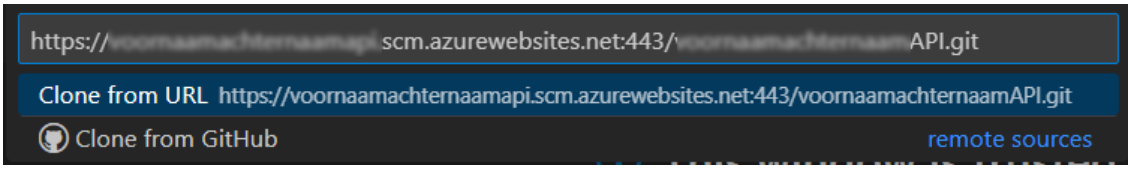
➔ Zorg dat het venster trusted is (Zodra dit niet het geval is, krijg je hier een melding van).

➔ Open een nieuw window via File → New Window

➔ Ga naar het tabblad "Source Control". Klik op Clone Repository:



➔ Er opent een zich een klein venster. Kopieer / plak je eerder gekregen Git Clone URI in. Druk daarna op Enter (of Clone from URL).

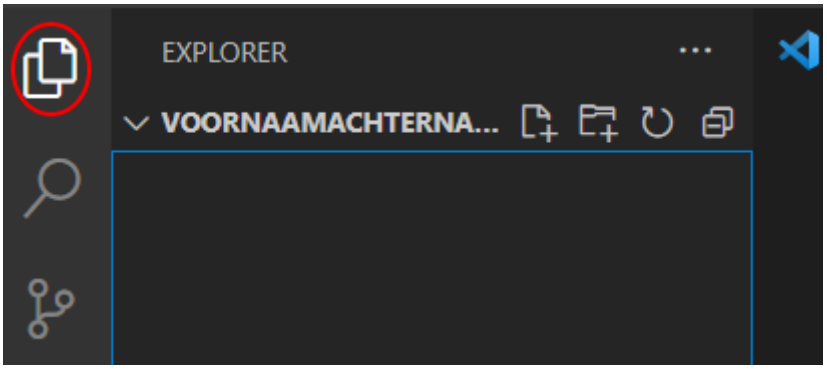


➔ Selecteer nu een makkelijk vindbare locatie waar de repository naartoe gaat.
Ons advies: Plaats deze in dezelfde map als waar ook Visual Studio Enterprise de repositories plaatst. Dat is de map:

C:\Users*JouwUser**source*|*repos*

➔ Open de zojuist ge-cloned repository:

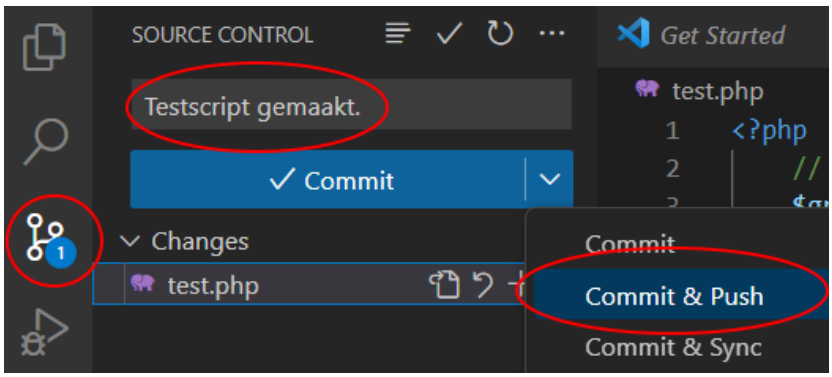
➔ Ga weer naar de explorer tab. Je dient nu een lege directory te zien (zoals in onderstaande afbeelding):



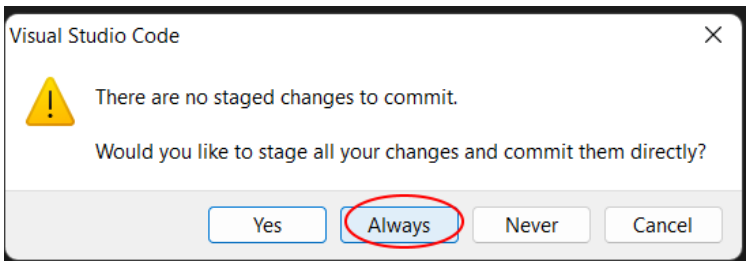
➔ Voeg hier een nieuw PHP-file aan toe genaamd "test.php".

➔ Schrijf een test-script die het e.e.a. aan HTML echo't. Dit om aan te tonen dat de zaak werkt. Zie onderstaand voorbeeld:

➔ Ga terug naar de source control tab en commit en push je gemaakt script. Vergeet niet een duidelijke commit message mee te geven.



➔ Vertel Visual Studio dat je altijd alle changes direct gestaged wilt hebben. Geen idee waarover dit gaat? Check dan de video over git binnen Visual Studio die je onderaan dit hoofdstuk kunt vinden.



Leer meer over Git / Source control binnen Visual Studio Code:
https://www.youtube.com/watch?v=i_23KUAEtUM&t=1s

Als de commit / push goed gegaan is, kunnen we gaan kijken of de code nu verstuurd is naar de Web App binnen Azure. Dit kunnen we doen via de webbrowser naar de URL van de Web App te gaan.

➔ Ga binnen Azure naar de Overview van de Web App: 

.

➔ Spannend moment: Bezoek de URL. Vergeet niet er /test.php achter te plaatsen.
Let op: De Web App is wellicht gaan "slapen" en kan enkele minuten duren om "wakker" te worden. Dit omdat wij de gratis variant gekozen hebben, deze is nogal lui 😊

- Je hebt zojuist een Web App opgezet en een eerste (test) HTML-applicatie hiervoor geschreven. In het volgende hoofdstuk gaan we leren hoe je nu de database gegevens in een JSON formaat kunt echo'n.

3. De API programmeren

Je hebt een Web App gebouwd, die als API-server zal fungeren. Nu is het tijd om de code te schrijven waarmee je de JSON string zal gaan echo'n richting de user. JSON is tegenwoordig dé standaard waarmee een API-server zijn data verstuurd naar de client.

📖 Wat is een JSON string ook alweer en wat kun je ermee? Bekijk het in deze korte video:

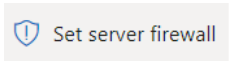
https://www.youtube.com/watch?v=7mj-p1Os6QA

3.1 Je API Client programmeren

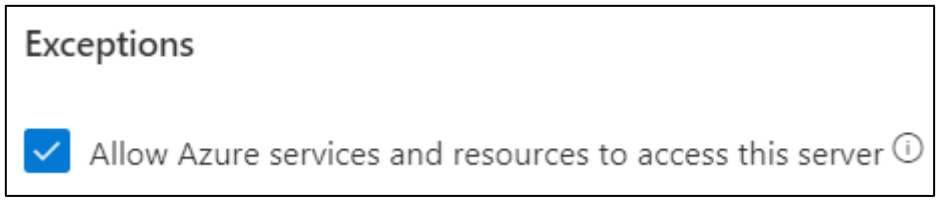
In deze reader hebben we ervoor gekozen om de code van de API in PHP te schrijven. Dit doen we om je even uit de C# bubbel te halen en weer te laten werken met een andere programmeertaal. Ook doen we dit, om je aan te tonen hoe meerdere programmeertalen kunnen samenwerken om één systeem te vormen.

Heeft de Web App toegang tot de SQL-server?

We gaan controleren of de Web App toegang heeft (via de Azure Firewall) tot de SQL-server. Doe daarom het volgende:

➔ Ga naar de Datatabase (Thema10) en klik op "Set server firewall". 

➔ Check of onderstaande vinkje aan staat:



➔ Sla dit op.

PHP-code om te connecten naar de database

Azure geeft ons gelukkig de PHP-code om naar een database te connecten al in de portal. Deze hoeven we alleen nog te kopiëren / plakken. Volg onderstaande stappen:

➔ Maak een nieuw bestand aan genaamd testAPI.php

➔ Per default staat de error reporting van Azure uit. Je krijgt dus geen errors/warnings te zien. Dit is onhandig. Zet onderstaande code bovenaa het PHP-script om error reporting aan te zetten:

```php
// Azure turns the error reporting OFF
// So we need to turn it ON again
ini_set('display_error', 1);
ini_set('display_startup_errors', 1);
error_report(E_ALL);
```

➔ Ga nu in de Azure portal naar je aangemaakte SQL database (Thema10).

➔ Ga naar het menuitem Connection strings 

➔ Hier vindt je de Connection string(s) / codevoorbeelden voor allerlei programmeertalen (bekijk ze maar eens).

➔ Klik op het tabblad PHP om de voorbeeldcode voor PHP te zien. 

➔ Je ziet hier meerdere voorbeelden van voorbeeldcode. Wij gaan de SQL Server Extension gebruiken. Neem deze PHP-code over.

```php
// SQL Server Extension Sample Code:
$connectionInfo = array("UID" => "ronspierings", "pwd" => "{your_password_here}",
"Database" => "Thema10", "LoginTimeout" => 30, "Encrypt" => 1, "TrustServerCertifi
cate" => 0);
$serverName = "tcp:voornaamachternaamserver.database.windows.net,1433";
```

➔ Verander je *UID* (Username), *pwd* (Password) en Database naar de gegevens waarmee jij kunt inloggen op de SQL-database (De gegevens van de database genaamd Thema10).

➔ Plaats daarna deze code in de PHP-file.

➔ Programmeer een try/catch om eventuele errors op te vangen en af te handelen:

```php
try {

}
catch(Exception $e)
{
    // Er is een fout opgetreden, waarschijnlijk door de database
}
```

➔ Plaats binnen de try de volgende code om een SQL-query uit te voeren en het resultaat in een array op te slaan. Daarna gebruiken we json_encode() om deze array aan de gebruiker te echo'n:

```php
try {
    $conn = sqlsrv_connect($serverName, $connectionInfo);

    // SQL query
    $query = "SELECT * FROM Oefening21";

    // Prepare the statement
    $result = sqlsrv_query($conn, $query);

    // Convert result to a Array
    $apiResult = [];

    // Loop through every row
    while($row = sqlsrv_fetch_array($result, SQLSRV_FETCH_ASSOC))
    {
        // Add the row to the existing Array
        array_push($apiResult, $row);
    }
}
catch(Exception $e)
{
    // Er is een fout opgetreden, waarschijnlijk door de database
    $apiResult["error"] = "Er is een onbekende fout opgetreden. Probeer opnieuw";
    $apiResult["message"] = $e->message();
}


// Show the apiResult as a JSON string to the user
echo json_encode($apiResult);
```

➔ Neem de bovenstaande code regel voor regel door. Beantwoord voor je zelf de volgende vragen

- Wat is de SQL-query die je uitvoert?
- Wat doet sqlsrv_query()?
- Wat doet sqlsrv_fetch_array() ?
- Wat doet array_push() ?
- Wat doet echo json_encode() ?
- Wat gebeurt er in het geval van een error?

➔ Hieronder zie je een overzicht van de code:

```php
// Azure turns the error reporting OFF
// So we need to turn it ON again
ini_set('display_error', 1);
ini_set('display_startup_errors', 1);
error_report(E_ALL);

// SQL Server Extension Sample Code:
$connectionInfo = array("UID" => "ronspierings", "pwd" => "P@ssword", "Database" => "Thema10", "LoginTimeout" => 30, "Encrypt" => 1, "TrustServerCertificate" => 0);
$serverName = "tcp:voornaamachternaamserver.database.windows.net,1433";

try {
    $conn = sqlsrv_connect($serverName, $connectionInfo);

    // SQL query
    $query = "SELECT * FROM Oefening21";

    // Prepare the statement
    $result = sqlsrv_query($conn, $query);

    // Convert result to a Array
    $apiResult = [];

    // Loop through every row
    while($row = sqlsrv_fetch_array($result, SQLSRV_FETCH_ASSOC))
    {
        // Add the row to the existing Array
        array_push($apiResult, $row);
    }
}
catch(Exception $e)
{
    // Er is een fout opgetreden, waarschijnlijk door de database
    $apiResult["error"] = "Er is een onbekende fout opgetreden. Probeer opnieuw";
    $apiResult["message"] = $e->message();
}

// Show the apiResult as a JSON string to the user
echo json_encode($apiResult);
```

➔ Commit & push de code. Test nu de API, door naar de URL te gaan.

3.2 Veel voorkomende problemen

404 not found

Krijg je een 404 error als je de website bezoekt? Maar weet je 100% zeker dat je URL klopt (ook hoofdletters gecontroleerd?):

- Is de commit/push wel gelukt? Controleer dit door in je Web App naar Deployment Center te gaan → LOG. Hier hoor je de commits / pushes te zien.
- Fout in je PHP-code. De meest voorkomende reden van een 404-error is het hebben van een error in de PHP-code. Los dit op door error reporting aan te zetten (zie hierboven) en je code nogmaals controleren.
-

Maak nu Taak 3 van van **oefening 3.1** om je eerder gemaakte API uit te lezen.

5. Security: API tokens

Deze uitbreiding op de reader komt z.s.m.