|  |
| --- |
| Educom|Detacom |
| Testcase software development v.1.2. |
| Project EducomOpdracht |

|  |
| --- |
| Justin Muris  6-24-2021 |

Contents

[Opdracht 2](#_Toc75452992)

[Stappenplan 3](#_Toc75452993)

[Algemeen idee 4](#_Toc75452994)

[Installatie 4](#_Toc75452995)

[Eventuele problemen/oplossingen 6](#_Toc75452996)

# Opdracht

Buienradar.nl stelt een API-koppelvlak ter beschikking van derden. Deze API kan geraadpleegd worden via **<https://data.buienradar.nl/1.1/feed/json>.**

* Bouw twee afzonderlijke applicaties:
  + De eerste communiceert periodiek, via een configuratiebestand in te stellen, met de API van Buienradar. De opgehaalde gegevens worden in een lokale database opgeslagen. Deze serverapplicatie heeft een specifiek API endpoint dat door de tweede applicatie geraadpleegd wordt. De API kent drie parameters:
    - Het betreffende weerstation;
    - De startdatum van de periode die we willen raadplegen;
    - De einddatum van de periode, bij default 7 dagen later dan de startdatum.
  + De tweede applicatie toont vervolgens een grafiek met de geselecteerde meeteenheden. Deze meeteenheden kunnen via checkboxes geselecteerd worden en zijn de meeteenheden zoals ze in de buienradar-API beschreven worden.

Voor de opdracht mag zelf bepaald worden waarin het ontwikkeld wordt, ook is een combinatie van programmeertalen toegestaan. Wees vooral creatief.

Het project wordt in een private repository opgeleverd en vervolgens gedeeld met “ReneKrewinkel” op GitHub.

Een werkende online versie met een goede gebruikershandleiding is een pre.

# Stappenplan

Als eerste stap heb ik een stappenplan opgesteld, om een soort van planning op te kunnen bouwen over de twee weken heen.

1. Creëer een Visual Studio solution met drie projecten erin.
   1. EducomOpdracht, ASP.NET Core Web App (Model-View-Controller);
   2. EducomOpdrachtAPI, ASP.NET Core Web API;
   3. EducomOpdrachtTaskScheduler, Console Application.
2. Maak een GitHub repository aan en upload deze (initial commit).
3. Eerst wordt de API in zijn basis opgezet, dit houdt in:
   1. GET-functie – haalt alles in de database op;
   2. POST-functie – plaatst een entry in de database;
   3. PUT-functie – updatet een entry in de database.
4. Als tweede wordt de console application gebouwd, dat als trigger wordt gebruikt om de database van de API te updaten met data uit de feed van Buienradar. Het enige dat de console applicatie doet is de API pingen/triggeren zodat het de functie uitvoert.   
   De console application wordt dan later ingesteld om periodiek uitgevoerd te worden.
5. Nu wordt de API verder uitgewerkt terwijl over tijd de database gevuld wordt met meer data uit Buienradar:
   1. Specifieke endpoints voor de GET-functie, een mét start- en eindperiode en een met alleen een startperiode;
   2. Een redirect van POST naar PUT zodra bekend is dat een entry al bestaat met dezelfde ID en/of datum;
   3. Een DELETE/wipe functie voor testing.
6. Wanneer de API gereed is, wordt aan de site gewerkt:
   1. Verbinding en dataretrieval met API;
   2. Data verwerken naar Model voor grafiekplotting;
   3. Grafiekplotting op basis van geselecteerde data;
   4. Meeteenheden togglen/filtreren.
7. Testen of alles goed werkt en een beetje idiot-proof is.
8. Achteraf wordt gedocumenteerd wat gedaan is, wat er nog eventueel gedaan moet worden en wat eventuele problemen kunnen zijn.
9. Proberen te hosten, wat waarschijnlijk niet goed lukt want ik ben geen server-persoon.

# Algemeen idee

De website toont de grafiek. De API levert data aan de website vanuit zijn database. De API verzamelt data uit de Buienradar-feed en verwerkt en plaatst deze in zijn database. De console applicatie triggert de API om data te verzamelen, want het is niet echt mogelijk om een periodieke task uit te voeren op een MVC pagina, en ook al zou het kunnen, is het mogelijk dat MVC of IIS gedurende een thread de Task besluit te droppen, waardoor het niet uitgevoerd wordt. Het is dus niet erg betrouwbaar.  
De console applicatie wordt periodiek uitgevoerd door middel van Windows Task Scheduler, en de interval wordt hier ook geconfigureerd.

# Installatie

Het project gebruikt veel externe tools en frameworks en dergelijke, dus er wordt veel gedownload.

1. Download en installeer de dotnet hosting bundle installer: <https://dotnet.microsoft.com/download/dotnet/thank-you/runtime-aspnetcore-5.0.4-windows-hosting-bundle-installer>
   1. Dit bevat allerlei frameworks en libraries die nodig zijn voor het hosten van de server.
2. Download SQL Express 2019 <https://www.microsoft.com/en-us/Download/details.aspx?id=101064> en SSMS <https://aka.ms/ssmsfullsetup>.
   1. Dit zal de primaire database zijn inclusief een management-tool.
3. Installeer SQL Express 2019, gebruik de Basic profiel. Laat alle instellingen op standaard staan. Klik op het eind op ‘Connect Now’ om te testen of het werkt. Als het goed werkt, verschijnt een CMD venster met ‘1 row affected’ onderaan.
4. Installeer SSMS, houd alle instellingen standaard.  
   Eenmaal opgestart, verbind met ‘[COMPUTER]\SQLEXPRESS’ via   
   Windows Authentication. Dit staat waarschijnlijk allemaal al vooringesteld.
5. Creëer een database door links met de rechtermuisknop op Databases te klikken en dan ‘New database...’. Vul als database naam ‘EducomOpdrachtAPI’ in, en klik op OK.
6. Open het bijgeleverde ‘SQL Script.sql’ bestand. Dit zou moeten openen in SSMS. Voer deze uit door op F5 te drukken. Als goed is krijg je geen errors terug en staan er nu twee nieuwe Tables in de database.
7. Houd de Windows knop en R in, zodat het venster Uitvoeren verschijnt. Type hier ‘optionalfeatures’ in, zodat je het venster Windows-onderdelen ziet. Klik op de checkbox bij ‘Internet Information Services’. Open vervolgens de map en ga naar ‘World Wide Web-services’, ‘Toepassingontwikkelingsfuncties’ en klik op de checkboxes voor ‘ASP’, ‘ASP.NET 3.5’ en ‘ASP.NET 4.8’. Klik op OK en laat Windows zijn werk doen.
8. Wanneer het klaar is, houd de Windows knop en R in voor het venster Uitvoeren en type hier ‘InetMgr’ in. Klap links de folder uit en klik met de rechtermuisknop op ‘Sites’ en vervolgens op ‘Website toevoegen...’. Vul een sitenaam in, dit maakt niets uit. Ik heb zelf gekozen voor ‘Educom Opdracht Site.  
   Kies een fysiek pad om de server in op te slaan. Zet de binding-type op https en kies onderaan een SSL-certificaat (de IIS Express Development Certificate is prima).  
   Laat de rest van de instellingen standaard en klik op OK. Herhaal dit voor de andere API, maar zet de port één verder en kies natuurlijk een andere sitenaam.
9. Open de Visual Studio solution met de drie projecten. Klik met de rechtermuisknop op EducomOpdracht en klik op ‘Publish’. Kies als ‘Publish method’ File System, of de optie die je gewoon laat kiezen in welke folder het komt te staan. Kies als Target location het fysiek pad dat je eerder hebt gebruikt om de site in op te slaan. Zet de configuration op Release en laat de rest standaard staan. Klik na de setup boven op ‘Publish’ als het dit nog niet vanzelf heeft gedaan.  
   Herhaal dit voor de API.
10. Ga terug naar IIS en ga naar Toepassingsgroepen. Klik op Educom Opdracht API en klik op Geavanceerde instellingen. Zet bij Procesmodel de ID op LocalSystem. Herhaal dit voor de Site. Ga terug naar de Sites en zorg ervoor dat de twee sites aanstaan door met de linkermuisknop erop te klikken en dan rechts op ‘Starten’ te klikken.
11. ‘Publish’ de console application op dezelfde wijze, alleen dan niet in de IIS folder, via ‘Folder’ en dan weer ‘Folder’. Sla deze op een andere veilige, permanente plek op. Klik na de setup boven op ‘Publish’ als het dit nog niet vanzelf heeft gedaan.
12. Houd de Windows knop en R in voor het venster Uitvoeren en type hier ‘taskschd.msc’ in. Maak links een nieuwe map aan. Klik dan met de rechtermuisknop op de map en klik op ‘Taak maken’. Voer een naam en beschrijving in, en klik dan op de Triggers tab en ‘Nieuw’. Stel het als volgt in:
    1. Start deze taak: Gepland
    2. Dagelijks, zet de start op een (toekomstig) rond uur zoals 14:00:00
    3. Taak herhalen elke: 1 uur – gedurende: oneindig
    4. Klik op OK.
13. Klik op Acties en klik op ‘Nieuw’. Klik op ‘Bladeren’ en navigeer naar de executable van de console applicatie die eerder gepublishet is. Klik op OK.
14. Ga naar de console applicatie die eerder gepublishet is. Zorg ervoor dat de ‘EducomOpdrachtTaskScheduler.dll.config’ keys als volgt uit zien:
    1. weerstationsUrl – <https://localhost:API_PORT/api/Weerstations/list>
    2. weerberichtenUrl – <https://localhost:API_PORT/api/Weerberichten/list>
       1. API\_POORT zou iets van 444 moeten zijn als je de stappen gevolgd hebt.
       2. Check dit ook voor de website in ‘app.config’!!
    3. consoleUsername – externalConsoleApp (mag veranderd worden)
    4. consolePassword – externalConsoleAppPassword (mag veranderd worden)
15. Ga dan naar de folder van de API en open daar ‘app.config’ en ‘web.config’, en zorg dat de consoleUsername en consolePassword overeenkomen met bovenstaand.
16. Open vervolgens ‘appsettings.json’ en zorg dat de SqlConnection als volgt staat:
    1. Server=localhost\\SQLEXPRESS; Database=EducomOpdrachtAPI; Initial Catalog=EducomOpdrachtAPI; Trusted\_Connection=True; Integrated Security=SSPI;

Als goed is staat nu alles op en zou het moeten werken. Als iets niet werkt, of als iets onduidelijk is, stuur mij een mail naar [justinmuris@gmail.com](mailto:justinmuris@gmail.com) en ik zal er naar kijken!

# Gebruikte NuGet packages/libraries

Newtonsoft.Json: gebruikt voor de API en de website zelf om data te transporteren. Het model van bijvoorbeeld de grafiek wordt naar json verwerkt en overgezet naar de JavaScript code op de pagina.

System.Configuration.ConfigurationManager: staat toegang toe tot de app.config en web.config bestanden, waarin onder andere login-informatie en connection strings zijn opgeslagen.

Microsoft.AspNet.Mvc: bevat wat extra extensions die nuttig waren om bepaalde HtmlActions uit te voeren op de site.

Microsoft.EntityFrameworkCore.\*: verschillende packages voor verschillende Entity Framework functies, die de database-toegang mogelijk maken op een simpele manier.

# Eventuele problemen/oplossingen

De Buienradar-feed slaat niks van het verleden op: alleen actueel weer of de meerdaagse voorspelling is beschikbaar. Verder dan vijf dagen vooruit wordt dus niet opgehaald, en niks uit het verleden wordt opgehaald. Daarom is het belangrijk om de console applicatie een dag of twee te laten lopen om fatsoenlijke resultaten te krijgen.

Visueel ziet de site niet erg speciaal of goed uit. Ik ben helaas geen grafisch designer, en ik heb mijn best gedaan. Bij mij geldt functie boven vorm, en dat kun je ook zien.

Ik weet niet of de database een upper-limit heeft op entries. Ieder half-uur heeft Buienradar weer nieuwe data van de weerstations, en dat geldt x51. Ieder half uur komen dus 51 nieuwe entries in de database. Als er eventueel een upper-limit blijkt te zijn, zou er een nieuwe methode moeten toegevoegd worden dat eerst de oudste 51 entries verwijdert voordat de nieuwe worden toegevoegd.

Een goed deel van mijn methoden zijn wellicht een beetje hackerig, maar het werkt. Hoe lang het werkt en hoe schaalbaar het is, ik heb geen idee. Er zijn vast ook makkelijkere en efficiëntere manieren om bepaalde methoden te herschrijven, maar daar ben ik mij niet van bewust op het moment. Ik heb al erg lang niet met een API gewerkt, en al helemaal niet met andermans API/feed, dus dit was compleet nieuw voor mij. Alsnog vind ik dat ik nog redelijk goed werk heb kunnen verrichten, ondanks mijn gebrek aan ervaring en roestigheid.

Ik heb ook niet meer gedacht aan checks of de API werkend is voordat de console applicatie of website ermee probeert te verbinden, wat ook voor errors kan zorgen.