VERSLAG WEBAPP

GEAVANCEERDE WEBAPPLICATIES

JARNE VAN DER PLAS
VRIJE UNIVERSITEIT BRUSSEL

Inhoudstafel

Inleiding

In de lessen geavanceerde webapplicaties zagen we alle verschillende elementen om een webapplicatie op te zetten. Tijdens de lessen maakten we een webapplicatie om de lampen in het lokaal aan, uit of op een specifieke waarde te zetten. Ook worden alle lampwaarden opgehaald wanneer iemand anders een waarde aanpast.
De webapplicatie heeft een backend die steunt op sqlite, gecreëerd met behulp van SQLAlchemy in Flask. Flask is een webframework dat een toevoeging biedt aan python. Python, flask en SQLAlchemy/Sqlite zorgen voor de backend. HTML, CSS en Javascript zorgen voor de front-end. Ook voegde ik een dockerfile, requirement, log files en readme toe aan mijn applicatie en voegde alles samen aan mijn github. Om de connectie met de lampen te verzorgen, wordt er een VPN opgesteld. Dit is nodig omdat het VUBNext netwerk met het IPv4 protocol werkt terwijl de modules gebruik maken van IPv6.

Front-end

De front-end is het onderdeel van de webapplicatie waar de gebruiker mee in contact komt. Dit zijn alle visuele elementen maar ook de invoervelden en het gebruiksgemak. Ook wordt er geprobeerd om de webpagina zo vlot mogelijk te laden.
De visuele elementen van de webpagina zijn hier het grootste onderdeel van. Elementen zoals de titel van de webpagina en de favicon zijn andere, niet te vergeten onderdelen.

HTML

HyperText Markup Language wordt gebruikt om een website op te maken. Je kan verschillende html files toevoegen zodat er verschillende templates ingeladen kunnen worden. Ik creëerde zeven verschillende html files. Hierin linkte ik de favicon, website titel, styles, een referentie naar bootstrap, een referentie naar verschillende css files en vooral de inhoud van de verschillende pagina's. Elke pagina heeft een navigatiebar. De inhoud van de navigatiebar verschilt van pagina tot pagina. Logischerwijs verschilt de inhoud ook per pagina.
Een interessante feature van HTML is dat if-statements aanvaard worden. Zo kon ik voor eenzelfde webpagina twee verschillende inhouden weergeven. Dit gebruikte ik bij mijn index pagina. Indien de gebruiker ingelogd is, kan die doorklikken. Indien de gebruiker niet ingelogd is, wordt er gevraagd in te loggen of een account te registreren. Zonder CSS zien webpagina's er heel oud en stijlloos uit.

CSS

Cascading Style Sheets wordt gebruikt om het opgemaakte HTML document mooier te maken. Er wordt een stijl toegevoegd aan de pagina. Deze kan door de gebruiker zelf worden opgebouwd. Ook bestaan er frameworks. Deze frameworks bevatten verschillende stijlen zodat de ontwikkelaar niet alles zelf moet declareren. Indien nodig, kan de gebruiker deze stijl altijd overschrijven of parallel aan het framework extra elementen declareren. In de webapp gebruikte ik bootstrap als framework. Ik gebruikte een voorbeeld op hun site als algemene richtlijn qua tekst, navigatiebar, slider etc.

Bootstrap

Bootstrap is een open-source framework gecreëerd door een groep ontwikkelaars bij twitter. Door een intern framework te maken, wouden ze de lay-out over de gehele (web)applicatie in eenzelfde lijn maken. Door het open-source te maken, en het dus open te stellen voor iedereen, werden er veel features toegevoegd door niet-twitter medewerkers. Ook is er een overzichtelijke documentatie te vinden. Soortgelijke frameworks zijn foundation en tailwind. Er zijn nog een hele hoop andere frameworks. Frameworks maken het eenvoudig om een mooie site te maken voor niet-front-end gerichte ontwikkelaars. Heel wat elementen zijn enkel nog toe te voegen door de correcte class te benoemen.

JavaScript

JavaScript wordt gebruikt om een script aan een html pagina toe te voegen. Door aan een html pagina <script> … </script> toe te voegen, wordt er een ruimte gemaakt om scripts uit te voeren. Deze scripts maken websites interactief. JavaScript kan ook in .js files geschreven worden zonder referentie naar een html pagina.
In de webapp om de lampen te besturen, wordt JavaScript gebruikt om de lamp waarde op te roepen. Ook wordt het gebruikt om de lampwaarde aan te passen wanneer een slider van waarde verandert. Ook wordt flask-socketio zowel in de JavaScript als in de backend python code toegevoegd.

jQuery, jQuery-Ajax

JQuery is een toevoeging aan JavaScript. Met behulp van Ajax, Asynchronous JavaScript and XML, is het eenvoudig om een connectie met een server op te zetten. Dit is zeer handig voor een webapp. Via Ajax kan eenvoudig een GET- of POST-request gestuurd worden. Er moet enkel een adres en in het geval van een POST-request ook een waarde meegeleverd worden. Vervolgens wordt de gewenste aanvraag gedaan. Zoals de naam verklapt, worden de aanvragen asynchroon uitgevoerd. Zo kunnen er verschillende aanvragen gebeuren vooraleer het eerste antwoord ontvangen wordt. Tijdens dat de daaropvolgende communicaties bezig zijn, kan de aanpassing al uitgevoerd worden op de webpagina. Indien om de X seconden de webpagina herladen wordt, is er meer belasting op de webserver. Ook kan noch de server, noch de webpagina in een meer energiezuinige modus gaan want beide blijven met elkaar communiceren onafhankelijk van aanvragen tot de lampen. Toch zullen beide methodes hetzelfde resultaat bekomen maar niet allebei even efficiënt.
De lampen worden met IoT (Internet of Things) devices bestuurd. De devices communiceren via een bridge naar de server. Op diezelfde server connecteert de webpagina. Via Ajax wordt een GET- of POST-request naar een meegeleverd webadres gestuurd. Indien er een succesvolle connectie was, wordt met een succesreactie geantwoord. Wanneer deze reactie ontvangen wordt, wordt de nodige code uitgevoerd. In het lampenvoorbeeld wordt bij een GET-request de lampwaarde opgevraagd en weergegeven op de webpagina. Indien een POST-request gestuurd wordt, wordt er gevraagd om de lamp waarde aan te passen. Vervolgens zal ook de lampwaarde op de webpagina aangepast worden. In tegenstelling tot de webpagina volledig te herladen, herlaadt Ajax maar een aantal elementen van de webpagina. In het voorbeeld worden enkel de waarden en de sliders aangepast.

3D Model Viewer

Een website kan interessanter gemaakt worden door het gebruik van een 3d-model. 3D-Modellen kunnen met tal van programma's gemaakt worden. Een gratis, zeer krachtige applicatie is blender. Deze modellen kunnen op verschillende manieren toegevoegd worden aan een website. De meest eenvoudige manier is via <model-viewer>. Na het toevoegen van het script, moet de .gltf of .glb file ingeladen worden op de gewenste locatie. Verschillende eigenschappen van het 3d model en de animatie kunnen gebruiksvriendelijk aangepast worden. Door het toevoegen van drie lijnen aan de html pagina, wordt een 3d-model weergegeven.

Backend

Met backend worden de onderdelen die niet zichtbaar zijn voor de gebruiker benoemd. Dit zijn de connecties met de webserver, laden van de pagina's, doorverwijzen etc. De processen zijn niet zichtbaar. In de backend test ik bijvoorbeeld of de aangevraagde lamp wel bestaat. Indien deze niet bestaat, filter ik het bericht. Ook wordt er nagekeken dat de aangevraagde waarde tussen nul en honderd ligt. Indien een kleinere waarde wordt aangevraagd, wordt de waarde op nul gezet. Wanneer te grote waarde aangevraagd wordt, wordt de waarde op honderd gezet. Dit is input parsing. Input parsing is belangrijk zodat de server niet overbelast wordt. Voor de gebruiker zijn deze acties niet zichtbaar.

Python-Flask

De backend wordt verzorgd door Python met flask als webframework. Flask bevat alle componenten om een website op te stellen. De applicatie wordt geïnstantieerd door app = flask.Flask(__name__) uit te voeren. Flask bezit heel wat interessante elementen voor de webapplicatie. Voor het inloggen bestaat er flask-login. Wanneer een gebruiker ingelogd is, zal de sessie onthouden worden door ingebouwde functies. Voor het bcrypten en hashen van een wachtwoord kan flask-bcrypt gebruikt worden. Met behulp van flask-socketIO kan een bidirectionele connectie tussen de webapplicatie en de webserver opgezet worden. Flask-Blueprint werd gebruikt om de login applicatie met de lampen applicatie te connecteren. Het moeilijke element is dat de lampen applicatie met socketio en asyncio werkt. Ook waren beide applicaties al parallel ontworpen.

Flask-Login

Het gebruik van flask-login zorgt voor heel wat handige features. Het registreren en inloggen wordt eenvoudig gemaakt door ingebouwde functies. Ook wordt er bijgehouden of de gebruiker ingelogd is. Indien een gebruiker niet ingelogd is, wordt die doorverwezen naar een unauthorized site. Dit kan ook eenvoudig aangegeven worden door @flask_login.login_required toe te voegen bij een functie. Ten slotte is het uitloggen ook eenvoudig gemaakt door de .logout_user() functie. Ook is er heel wat beveiliging ingebouwd.

Flask-Bcrypt

Wanneer een paswoord in een database wordt opgeslagen, willen we dit niet als de exacte tekst opslaan. Bcrypt zorgt ervoor dat het paswoord gehasht wordt. Dit houdt in dat een algoritme van een tekst, hier een paswoord, een onherkenbaar patroon maakt. Dit patroon wordt op de server opgeslagen. Wanneer iemand wil inloggen, zal het paswoord opnieuw gehasht worden en beide hashes worden vergeleken. Bcrypt bevat ook de functionaliteit om een wachtwoord te salten. Salten is het toevoegen van characters aan een tekst of paswoord vooraleer het gehasht wordt. De characters kunnen vanvoor of vanachter aan de tekst toegevoegd worden. De salt tekst wordt op de server opgeslagen zodat dit kan toegevoegd worden wanneer een account probeert in te loggen. Een salt is specifiek per account. Dit maakt het moeilijker voor hackers om de hash te achterhalen. Deze functies zijn ingebouwd in flask. Deze worden ook uitgevoerd bij het registeren en inloggen in de webapplicatie.

Flask-SocketIO

Flask-SocketIO zorgt bij de backend functionaliteit voor de bidirectionele connectie tussen de lampen en de server. Wanneer er een request bericht gestuurd wordt vanuit de html template, zal de server de lampwaarde via socketIO terug sturen. Nadien wordt de waarde en slider positie aangepast. De bidirectionele connectie zorgt ervoor dat de server zelf een bericht kan uitsturen naar de webpagina wanneer de merkt dat een van zijn waardes aangepast is, wanneer een event gebeurd is. Wanneer een andere gebruiker de lampwaarden aanpast, gaat dit via de server. De server is dus alwetend en deelt dit graag met de rest van de gebruikers.
Een andere oplossing zou zijn de webpagina elke X seconden te herladen. Dit kan echter een grote lading op de server en site plaatsen. Wanneer de lampen een tijd niet verandert zijn van waarde, is dit ook een waardeloze belasting op het netwerk. SocketIO belast het netwerk enkel wanneer nodig en met gerichte berichten.

Blueprint

Blueprint zorgt voor een duidelijke structuur zodat parallel ontwikkelen makkelijk wordt. Verschillende elementen kunnen los van elkaar ontwikkeld en getest worden vooraleer ze samen gebracht worden. Het zorgt ervoor dat er slechts één server opgestart moet worden voor verschillende los van elkaar ontwikkelde webapplicaties. Indien gewenst, kan er een parent-child hiërarchie opgezet worden. Dit kan dan gebruikt worden om error excepties slechts één keer te declareren voor een hoop child-pagina's.
Voor het samenbrengen van de log-in applicatie, die niet via socketIO wordt uitgevoerd, en de lampen applicatie, die wel via socketIO werkt, gebruikte ik blueprint. Blueprint zorgt voor een hiërarchische structuur in de applicatie. Door main = Blueprint('main', __name__) toe te voegen, kon ik overal socketio.route() of app.route() vervangen door main.route().
De blueprint wordt in __init__.py geïnitialiseerd en uitgevoerd in elke functie waar er een route wordt gecreëerd. In app.py wordt de blueprint geregistreerd. Dit wordt gedaan op het moment dat bcrypt, loginmanager en socketIO ook worden geïnitialiseerd en geregistreerd.

SQlite

SQlite is een databasesysteem dat gebruik maakt van SQL. Zoals de naam zegt, is het een lichtere versie van SQL. In SQL kan je met het gebruik van queries gegevens uit een database opvragen.
In de les en opgave gebruikten we geen queries. In plaats daarvan gebruiken we ORM, object-relational mapping. Met het gebruik van ingebouwde libraries, vragen we via de python code gegevens op. Er worden geen queries expliciet uitgeschreven. De libraries zijn een onderdeel van flask-sqlalchemy, wat een uitbreiding is voor flask.

Flask-SQLAlchemy

SQLAlchemy wordt in de code geïmporteerd en geïnstantieerd. Nadat deze is geïnstantieerd, wordt er een database aangemaakt op de gewenste locatie. De inhoud van de database is door de gebruiker zelf in te stellen. Hier kan deze aangeven of het een primary key is of het uniek moet zijn etc. Het voordeel is dat alles op een plaats gecentraliseerd is. De queries zijn te vinden bij de rest van de code. Veel onderdelen worden op de achtergrond uitgevoerd zonder dat de developer er veel

aandacht aan moet besteden. SQL is een zeer krachtige taal en tool, toch wordt dit niet altijd zo gezien. Door met ORM te werken, wordt de kracht behouden en moet de developer zich niet specialiseren in SQL. De developer kan zich in de gekende taal, hier python, verdiepen om een beter model te schrijven.

Connectie

De frontend en backend zijn geschreven en geconnecteerd, echter is er nog geen connectie naar de lampen. De lampen communiceren met een server via CoAP, constrained application protocol. In python bestaat de aiocoap bibliotheek als hulp voor de communicatie. Het protocol wordt asynchroon uitgevoerd. Dit betekent dat taken tegelijkertijd uitgevoerd kunnen worden. Dit is belangrijk voor de applicatie. Wanneer een bericht verzonden wordt, duurt het tijdje vooraleer er een antwoord wordt ontvangen. In deze tussentijd willen we nog andere taken kunnen uitvoeren. Wanneer het synchroon zou verlopen, zouden we moeten wachten op een antwoord vooraleer de volgende taak uit te voeren. Wat voor een heel traag systeem zou zorgen.

CoAP

Ingebouwd in de aiocoap bibliotheek in python zijn de functies om taken als protocol, request en response af te handelen. In protocol wordt het communicatieprotocol opgezet. In request wordt de manier van bericht doorgegeven. Coap gebruikt het RESTful model dus dit kan GET, PUT, POST of DELETE zijn. Vervolgens wordt ook de uri en indien het een PUT bericht is, de waarde doorgestuurd. Ten slotte proberen we een antwoord te ontvangen. Indien er niets wordt ontvangen, wordt een error weergegeven. Indien het antwoord correct werd ontvangen, wordt dit weergegeven. De data kan als een XML, JSON , CBOR of … doorgestuurd worden. Deze stappen gebeuren niet onmiddellijk. Er is een bepaalde tijd nodig om alle stappen af te ronden. Om deze rede is CoAP een asynchroon proces. We willen dat andere taken tegelijkertijd uitgevoerd kunnen worden. Een asynchroon proces neemt meer tijd in ten opzichte van een synchroon proces. Wanneer we met een server communiceren, wensen we een asynchroon proces. De communicatie is niet ogenblikkelijk dus willen we andere functies tegelijkertijd kunnen uitvoeren. Ook is de volgorde van het uitvoeren van de lampaanvraag minder belangrijk. Bij een synchroon proces is de volgorde vaak belangrijk.

JSON

JSON staat voor JavaScript Object Notation en wordt gebruikt om data gestructureerd te verzenden. Het is een gegevensformaat. Het voordeel van gestructureerde datatransmissie is dat de inhoud duidelijk is voor alle belanghebbenden. Ook kan er eenvoudig één element uit de transmissie opgevraagd worden. De structuur is eenvoudig om te lezen en te begrijpen dankzij de key-value-pairs. Een ander gegevensformaat is XML, Extensible Markup Language. Het grootste verschil is de manier waarop het geparset wordt. Ongeveer elke website bevat JavaScript waardoor het parsen van een JSON document eenvoudig gaat. Ook heeft JSON geen endtag. XML heeft dit wel, er is een soortgelijke structuur als een html document. Gegevens starten met <name> en eindigen met </name>. Bij de naam kunnen elementen en attributen meegegeven worden. De opmaak is afkomstig van SGML, Standard Generalized Markup Language. HTML is ook afgeleid uit SGML.

Vandaar de overeenkomstige opmaak. De <name> ..</name> opmaak maakt een XML file langer en minder leesbaar. XML kan net als JSON genest zijn. Eén lijst kan meerdere key-value-pairs bevatten.

RESTFul

CoAP gebruikt de RESTful, REpresentational State Transfer, architectuur om data via HTTP te communiceren. Er wordt door de server toegang geboden via een aantal manieren of poorten. Het is aan de gebruiker om deze poorten op de correcte manier aan te spreken. Dit door naar de correcte URI, Uniform Resource Identifiers , of URL, uniform resource locator te verzenden. Het RESTful-protocol bevat een aantal standaard HTTP-methodes: GET, POST, PUT, DELETE etc. Dit wordt gecombineerd met de data tot een JSON-structuur om zo verzonden te worden.

Logger Files

Logger files kunnen gecreëerd worden om bij te houden welke errors er gebeurd zijn, wat de console print, wat gebruikers hebben gedaan etc. Wanneer deze geanalyseerd worden, kan het patroon van de klant duidelijk worden. De structuur van een website kan aangepast worden zodat een klant minder knoppen moet induwen vooraleer deze bij de gewenste pagina komt. Ook fouten van de server of webpagina worden opgeslagen. Log files worden automatisch aangevuld. Wanneer een error ‘s nachts voordoet, kan dit in de ochtend opgelost worden.
In de webapplicatie opdracht maak ik automatisch twee log files aan. Eén log file houdt server berichten bij, de andere houdt het gedrag van de gebruiker bij. Er zijn maar een beperkt aantal handelingen die uitgevoerd kunnen worden op de webapplicatie. Voor al deze handelingen bestaan er berichten die in de log file worden opgeslagen.

VPN

Vooraleer de webapplicatie uitgevoerd kan worden, dient de laptop of server te connecteren met een VPN. Een VPN is een virtual private network. Er zijn verschillende soorten VPN’s: Remote access, site-to-site, extranet-based site-to-site. Het kan gebruikt worden binnen bedrijven om via een extern netwerk toch toegang te hebben tot interne gegevens. Ook kan het gebruikt worden om je locatie aan te passen. Wanneer iets geolocked is, kan je met een VPN je locatie aanpassen zodat je wel toegang krijgt tot de gewenste inhoud. In onze applicatie wordt het gebruikt om IPv6 berichten over een IPv4 netwerk te sturen. De modules werken volgens het IPv6 protocol terwijl het VUBnext netwerk op IPv4 werkt. De oplossing is om met een VPN te connecteren om toch het IPv6 protocol te kunnen volgen.

DockerFile

Een dockerfile is een lijst met instructies die uitgevoerd moeten worden zodat een applicatie in zijn geheel kan gebouwd worden. Applicaties hebben vaak een lijst van vereisten op vlak van bibliotheken die toegevoegd moeten worden zodat de applicatie kan werken. Meestal heeft een webapplicatie meer nodig dan enkel bibliotheken. Verschillende applicaties rondom de webapp moeten ook geïnstalleerd worden vooraleer het geheel uitgevoerd kan worden. Een database wordt

bijvoorbeeld gebruikt om accounts te registreren en op te slaan. Deze moet ook toegevoegd worden aan de applicatie. Ook wordt een start pad geregistreerd. Wanneer er verschillende folders zijn aangemaakt die naar elkaar verwijzen, moet het start pad duidelijk gemaakt zijn vooraleer de verwijzingen correct zijn. Ook is het handig om één instructie op te roepen en dat vervolg al het nodige automatisch geïnstalleerd en geïnitialiseerd wordt.
Voor dit geheel worden dockerfiles gemaakt. Een dockerfile bevat een hoop instructies en referenties die uitgevoerd kunnen worden. Wanneer deze zijn uitgevoerd, is alles klaar om de webapplicatie te starten.

Git

Git is een open-source software voor versiebeheer. Het is de basis voor applicaties als github en gitlab. Het slaat oudere versies van projecten op en laat toe deze te branchen, mergen en nieuwere versies toe te voegen. Github en gitlab zijn platformen die git als basis bouwsteen hebben gebriikt. Github is een cloud platform waar gebruikers hun code kunnen uploaden en waar vorige versies opgeslagen blijven. Dit maakt het mogelijk om naar een vorige versie terug te gaan. Ook maakt het samenwerken in teamverband eenvoudig door elementen als pull, branch, merge etc. Het is ook een plaats om persoonlijke code tentoon te stellen. Gitlab is een soortgelijk platform. Het werd tijdens de lessen gebruikt om de voorbeeldcode aan de studenten beschikbaar te stellen. Als student gebruik ik github om mijn code op te slaan en indien toegelaten tentoon te stellen.

Referenties

https://getbootstrap.com/

https://jquery.com/

https://api.jquery.com/jquery.ajax/

https://modelviewer.dev/

https://flask.palletsprojects.com/en/2.2.x/

https://flask-login.readthedocs.io/en/latest/

https://flask-bcrypt.readthedocs.io/en/1.0.1/

https://flask-socketio.readthedocs.io/en/latest/

https://socket.io/docs/v4/

https://docs.python.org/3/library/asyncio.html

https://flask.palletsprojects.com/en/2.2.x/blueprints/

https://sqlitebrowser.org/

https://flask-sqlalchemy.palletsprojects.com/en/3.0.x/

https://aiocoap.readthedocs.io/en/latest/index.html#

https://www.json.org/json-en.html

https://nl.wikipedia.org/wiki/Extensible_Markup_Language

https://en.wikipedia.org/wiki/Virtual_private_network

https://nl.wikipedia.org/wiki/Internet_Protocol_versie_6

https://docs.docker.com/engine/reference/builder/