

Geïntegreerde proef

Slimme parkeergarage

|  |  |
| --- | --- |
| Voornaam + Naam: | Bram Swinnen |
| Studierichting: | Applicatie- en Databeheer |
|  | Tweede leerjaar – 3de graad |
| Schooljaar: | 2024-2025 |
|  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tel. OV3: 011 79 93 30  Tel. OV4: 011 79 93 70 | Duinenstraat 1  BE-3920 Lommel | www.provilion.be  info@provilion.be |  | Afbeelding met logo  Automatisch gegenereerde beschrijving |



Geïntegreerde proef

Slimme parkeergarage

|  |  |
| --- | --- |
| Voornaam + Naam: | Bram Swinnen |
| Studierichting: | Applicatie- en Databeheer |
|  | Tweede leerjaar – 3de graad |
| Schooljaar: | 2024-2025 |
|  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tel. OV3: 011 79 93 30  Tel. OV4: 011 79 93 70 | Duinenstraat 1  BE-3920 Lommel | www.provilion.be  info@provilion.be |  | Afbeelding met logo  Automatisch gegenereerde beschrijving |

Woord vooraf

Ik heb ervoor gekozen om een slimme parkeergarage te ontwikkelen. De reden hiervoor is dat ik graag een project wilde maken dat ik me goed kon inbeelden en waarvan ik kon zien hoe het in de praktijk toegepast zou kunnen worden. Het idee van een slimme parkeergarage sprak me aan omdat het een concreet, realistisch concept is dat ook effectief uitgewerkt kan worden tot een volledig functioneel systeem.

Ik heb dit project volledig zelfstandig uitgewerkt. Dit was een bewuste keuze, omdat ik tijdens het proces geen directe hulp nodig had. Alles wat ik wilde weten of moest leren, kon ik zelf online opzoeken en toepassen. Op die manier heb ik niet alleen het project opgebouwd, maar ook mijn probleemoplossend vermogen en zelfredzaamheid versterkt.

Inhoudsopgave

Inleiding

Voor mijn GIP-project besloot ik een slimme parkeergarage te ontwikkelen, gericht op het verbeteren van de parkeerervaring voor zowel gebruikers als beheerders. Het doel was om een webapplicatie te creëren waarmee gebruikers eenvoudig parkeerplaatsen kunnen reserveren en in- en uitrijden via nummerplaatherkenning. Daarbij konden ze hun voertuigen beheren, reserveringen volgen en betalingen verrichten, terwijl de beheerders realtime gegevens kunnen inzien en beheren, zoals de bezetting van de parkeergarage en de inkomensinformatie.

De applicatie is ontworpen voor een breed publiek: van mensen die dagelijks een parkeerplaats nodig hebben tot bedrijven die hun parkeerfaciliteiten willen optimaliseren. De technologieën die ik gebruikte waren Flask voor de backend, React voor de frontend en een database die alle gegevens beheerde. Een belangrijke beperking in dit project was de tijdsdruk en het technische niveau, waardoor sommige geavanceerdere functies, zoals live nummerplaatherkenning, niet haalbaar waren. Desondanks heb ik geprobeerd de applicatie zo goed mogelijk te bouwen en te testen, en ik ben blij met het resultaat.

Het project is opgebouwd rond een gebruiksvriendelijke interface voor de eindgebruiker, terwijl de beheerder toegang heeft tot uitgebreide gegevens via een adminpagina. Door deze applicatie te bouwen, wilde ik niet alleen mijn programmeerkennis uitbreiden, maar ook een praktisch, schaalbaar product creëren dat direct toepasbaar is in de praktijk. In de volgende secties ga ik in detail in op de opbouw, de technische keuzes en de uitdagingen die ik ben tegengekomen tijdens de ontwikkeling van deze applicatie.

Eigenlijke tekst (eigen indeling)

**Inleiding**

Ik ben gestart met het brainstormen over een geschikt idee voor mijn GIP-project. Na het overwegen van enkele concepten, wilde ik iets ontwikkelen dat ook effectief toepasbaar is in het dagelijkse leven. Zo is het idee ontstaan voor mijn project: **de slimme parkeergarage**.

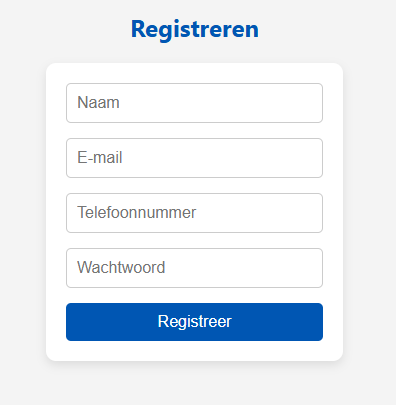
Het resultaat is een webapplicatie waarmee gebruikers eenvoudig parkeerplaatsen kunnen reserveren. Ze kunnen hun voertuigen registreren, reserveringen beheren en automatisch in- en uitrijden via nummerplaatherkenning. Alle informatie wordt overzichtelijk weergegeven en bijgehouden.

Daarnaast is er een aparte **adminpagina** voorzien voor het beheer van de parkeergarage. Daarop zijn onder meer de actuele bezetting, inkomgegevens en geplande reserveringen zichtbaar.

Afbeelding met tekst, diagram, lijn, Parallel

Door AI gegenereerde inhoud is mogelijk onjuist.**1. Projectopbouw – Parking Applicatie**

Ik ben begonnen met het opzetten van een eenvoudige website met een combinatie van **Flask** voor de backend en **React** voor de frontend. Daarna koppelde ik de database en stelde ik een klassendiagram op als basis voor de datastructuur.

Vervolgens heb ik een **authenticatiesysteem** ontwikkeld. Elke gebruiker kan een eigen account aanmaken, waarop zijn/haar voertuigen en reserveringen worden bijgehouden. Dit systeem gebruikt **tokens**, die bij het inloggen worden gegenereerd en bij elke navigatie worden meegegeven. Hierdoor blijft de sessie geldig en weet het systeem dat de gebruiker is ingelogd.

Omwille van privacy en veiligheid worden wachtwoorden **gehashed** opgeslagen in de database. Hier een voorbeeld van hoe dit er uit ziet.  De registratie laat slechts één account per e-mailadres toe, om verwarring te vermijden. Ik heb **Werkzeug** gebruikt voor alles dat met wachtwoorden te maken heeft. Dit systeem heb ik gekozen na een zoektocht naar een betrouwbare oplossing die goed integreerbaar is met Flask.

**2. Voertuigen registreren**

Na het opzetten van het gebruikerssysteem begon ik met het ontwikkelen van de voertuigregistratie. Elke gebruiker kan meerdere voertuigen opslaan in de applicatie. Dit maakt het mogelijk om reserveringen per voertuig op te volgen.

Via de knop **"Voertuigen"** navigeer je naar een overzicht. Nieuwe voertuigen kunnen worden toegevoegd via een formulier waarin je de volgende gegevens invult:

* Nummerplaat
* Grootte van het voertuig
* Of het voertuig elektrisch is
* Of het een plaats voor personen met een handicap nodig heeft



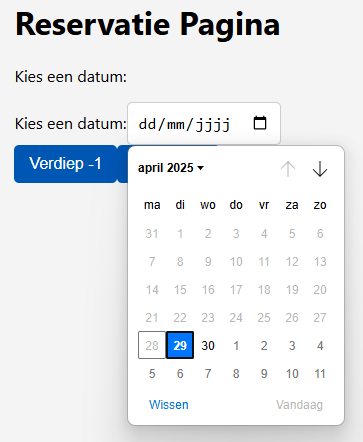
Deze gegevens worden bij het klikken op "Toevoegen" direct aan de database toegevoegd. Er wordt gecontroleerd op een **unieke nummerplaat**: als deze al bestaat, kan je deze niet opnieuw ingeven. Elk voertuig krijgt een eigen ID zodat het altijd makkelijk bereikbaar is. In de database wordt ook het **gebruiker\_id** opgeslagen, zodat voertuigen aan gebruikers gekoppeld blijven. Elke gebruiker kan dus meerdere voertuigen registreren.

**3. Reservatiesysteem**

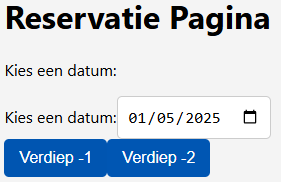
Daarna volgde het hart van de applicatie: **het reservatiesysteem**. Ik koos ervoor om de parkeergarage voor te stellen met twee verdiepingen, elk onderverdeeld in negen zones. Elke zone bevat 40 parkeerplaatsen.

Het reserveringsproces verloopt in stappen:

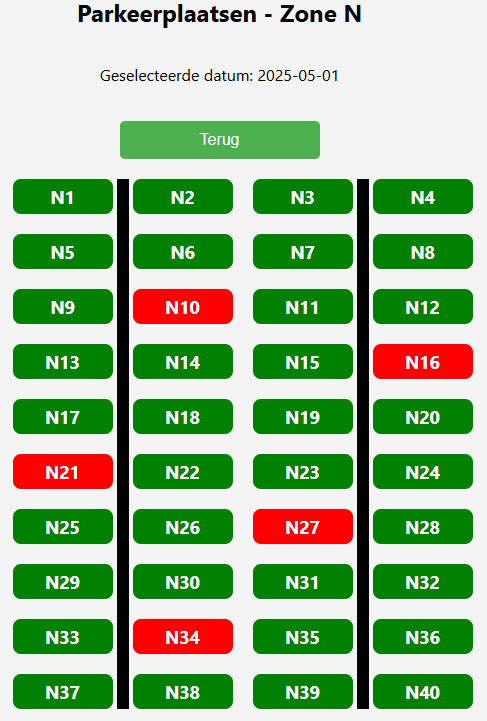
1. Kies een datum – alleen beschikbare plaatsen voor die dag worden getoond.



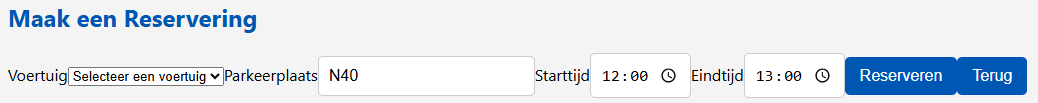
1. Selecteer een verdieping (-1 of -2).

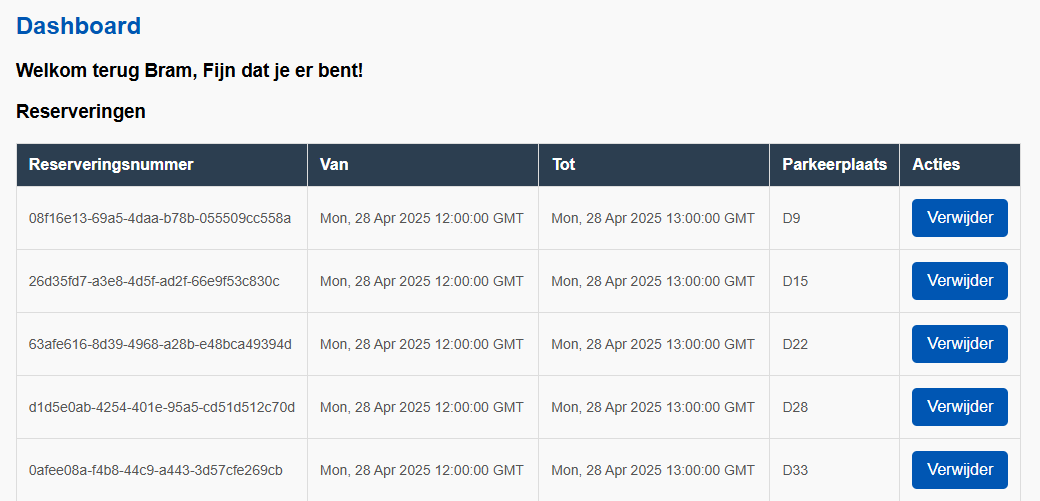


1. Kies een zone (alfabetisch gesorteerd). 
2. Beschikbare plaatsen worden **groen** weergegeven, bezette plaatsen **rood**.



Klik je op een vrije plaats, dan kom je op een scherm waar je je reservering voltooit door je nummerplaat en gewenste uur in te geven.

Na bevestigen krijg je een melding en word je doorgestuurd naar het **dashboard**. Hier zie je al je actieve en toekomstige reserveringen.

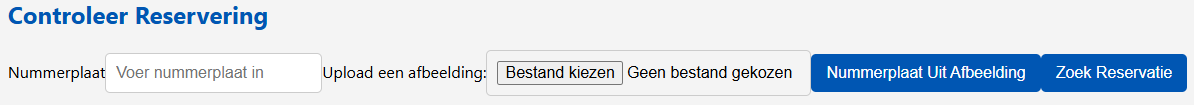


Je ontvangt ook een bevestigingsmail met alle details. Reserveringen kunnen eenvoudig worden geannuleerd.

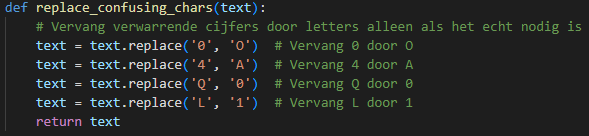
Er is een beperking ingebouwd zodat reserveringen minstens **één dag op voorhand** moeten gebeuren. Dubbele reserveringen worden vermeden doordat het systeem bij elke reservatie controleert of de gekozen plaats al bezet is. Bezette plaatsen worden rood gekleurd en kunnen niet aangeklikt worden.

**4. In- en uitrijsysteem**

Het volgende onderdeel is het **systeem voor in- en uitrijden** van voertuigen op basis van **nummerplaatherkenning**.

Bij aankomst wordt de nummerplaat herkend via een geautomatiseerd systeem. Als er een geldige reservering is, wordt getoond waar je mag parkeren.  


Aanvankelijk gebruikte ik hiervoor **OpenCV**, met filters zoals grijswaarden en verscherping. Dit bleek echter foutgevoelig, dus schakelde ik over naar **OCR-technologie** via **Pytesseract** (een wrapper rond Tesseract-OCR). Dit maakt het mogelijk om tekst uit beelden te halen.

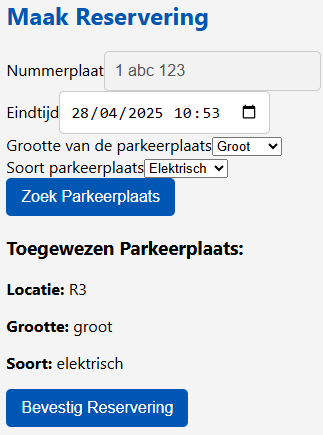
Voor de herkenning verwijder ik eerst alle speciale karakters uit de afbeelding. Daarna vergelijkt het systeem de herkende tekst met gekende nummerplaatformaten. Bij verwarring corrigeert het systeem zelf de karakters en voert de herkenning opnieuw uit.

De herkende nummerplaat wordt in een invoerveld getoond, zodat je manueel kan aanpassen. Daarna klik je op **"Zoek reservering"**. Als er een reservering is, worden de gegevens getoond. Zo niet, kan je kiezen om alsnog te reserveren of te annuleren.

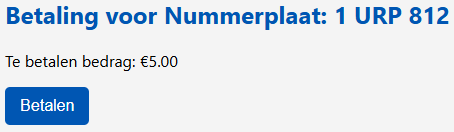
Bij een nieuwe reservatie geef je volgende info op:

* Of het voertuig elektrisch is
* Of je een gehandicaptenplaats nodig hebt
* De grootte van het voertuig
* Het gewenste einduur

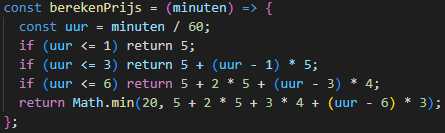
Het systeem zoekt op basis hiervan automatisch een geschikte plaats.



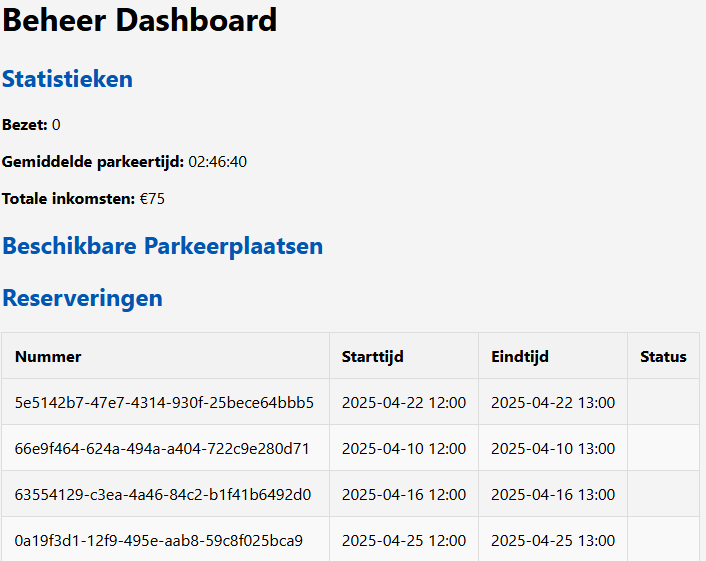
**5. Uitrijden en betaling**

Het uitrijsysteem werkt op gelijkaardige wijze. De nummerplaat wordt manueel ingevoerd. Vervolgens word je doorgestuurd naar een **betaalpagina**, waar het te betalen bedrag kan zien.

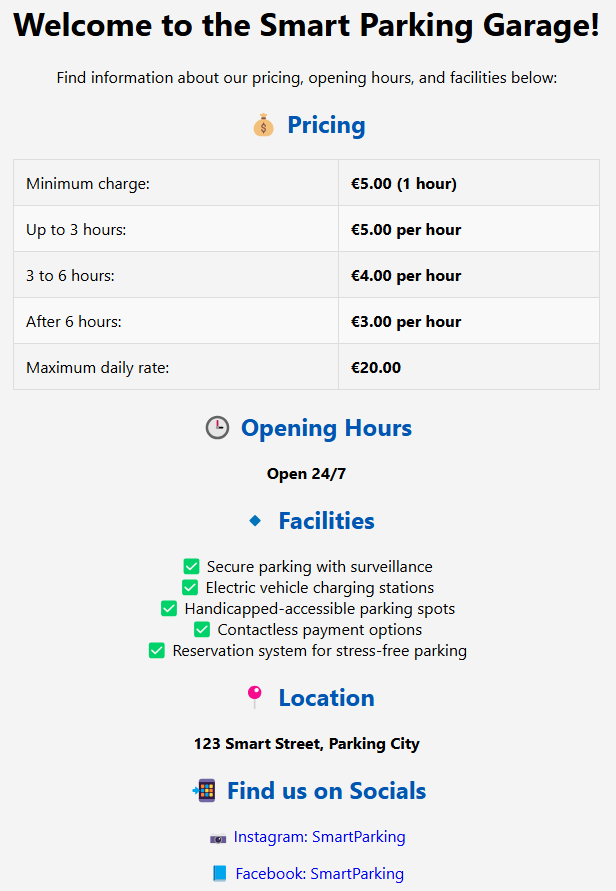
Bij het klikken op **"Betalen"** wordt de betaling gesimuleerd, aangezien het project niet gekoppeld is aan een echte betalingsprovider.

Het bedrag wordt berekend via een simpele functie die kijkt hoe lang je geparkeerd stond. De prijs wordt dan berekend op basis van een **vast tarief per uur**.

**6. Adminpagina en afwerking**

Voor het beheer ontwikkelde ik een **adminpagina** met o.a.:

* Real-time overzicht van bezette plaatsen
* Dagelijkse inkomsten
* Lijst van toekomstige reserveringen



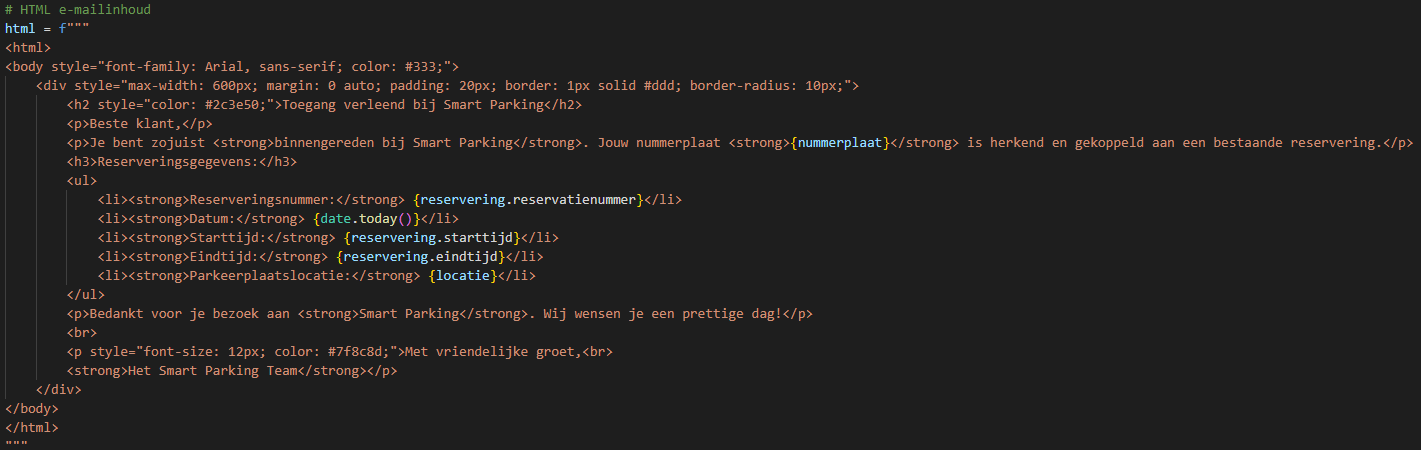
Tot slot werkte ik de applicatie af met een overzichtelijke **startpagina**. Hierop staat alle nuttige informatie: openingsuren, tarieven, en extra uitleg over het systeem. Ik heb gekozen voor een uniforme kleurenstijl zodat de applicatie er modern en professioneel uitziet. Alle onderdelen vormen samen een **visueel en functioneel geheel**.

**7. Mailsysteem**

Ik heb ervoor gekozen om in mijn applicatie ook een mailsysteem te integreren. Dit zorgt ervoor dat het geheel niet alleen realistischer aanvoelt, maar ook gebruiksvriendelijker is voor de eindgebruiker. De mails bieden duidelijke bevestiging bij belangrijke acties zoals registratie, reserveringen of annulaties, wat in echte toepassingen ook verwacht wordt.

Voor de technische uitwerking heb ik gebruikgemaakt van een **SMTP-server**, in dit geval van Gmail. Ik koos voor Gmail omdat ik hiermee al ervaring had opgedaan tijdens mijn stage, waar ik ook een project ontwikkelde waarbij automatische e-mails verstuurd moesten worden. Gmail bleek daar de meest gebruiksvriendelijke en toegankelijke optie.

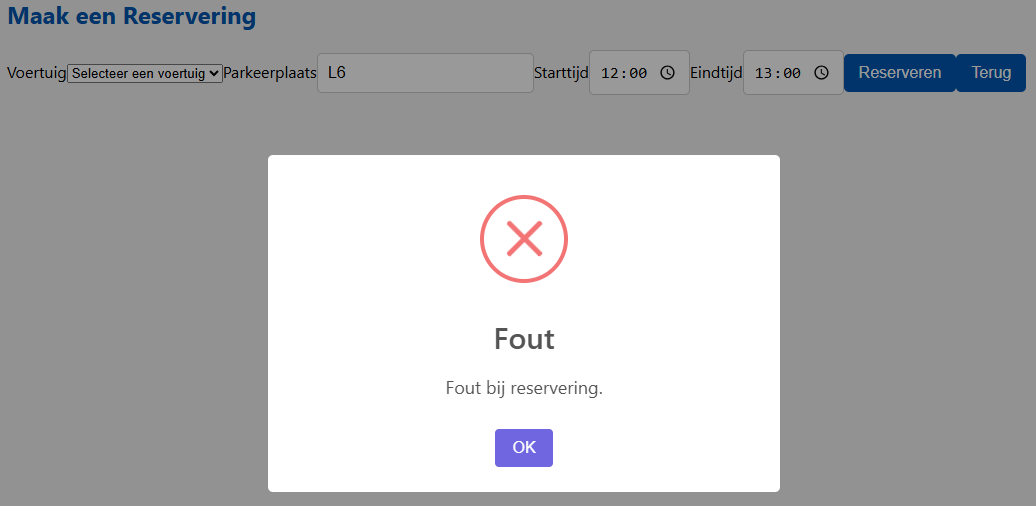
In mijn toepassing worden alle mails verstuurd via mijn persoonlijke e-mailadres naar het adres van de betreffende gebruiker. Elke soort mail (bijvoorbeeld: registratiebevestiging, reserveringsbevestiging of annuleringsmail) heeft een **eigen HTML-sjabloon**.

Hierin zitten telkens een aantal vaste elementen, zoals een standaard layout en opmaak, aangevuld met dynamische gegevens die betrekking hebben op de specifieke situatie. Zo is elke mail inhoudelijk duidelijk en professioneel opgesteld.

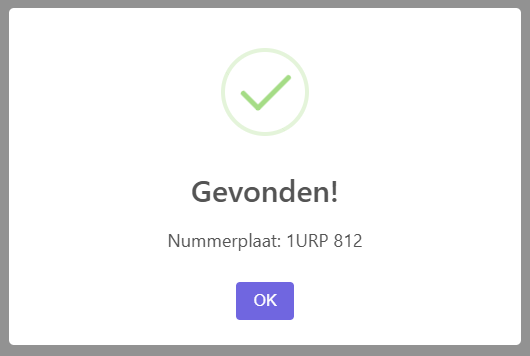
Om Gmail veilig te kunnen gebruiken met externe toepassingen, moest ik een **app-specifiek wachtwoord** aanmaken. Dit is een wachtwoord dat je kunt genereren nadat je **tweestapsverificatie** hebt geactiveerd op je Google-account. Hiermee geef je een externe applicatie (in mijn geval mijn Python-code) toegang tot je e-mail, zonder je echte wachtwoord te moeten delen. Na het instellen van 2FA (two-factor authentication) heb ik via de Google-instellingen dit app-wachtwoord aangemaakt, wat relatief eenvoudig verliep.

Het resultaat is een robuust mailsysteem dat betrouwbaar werkt en ervoor zorgt dat gebruikers altijd op de hoogte zijn van hun acties binnen de applicatie.

**8. Meldingen voor gebruikers**

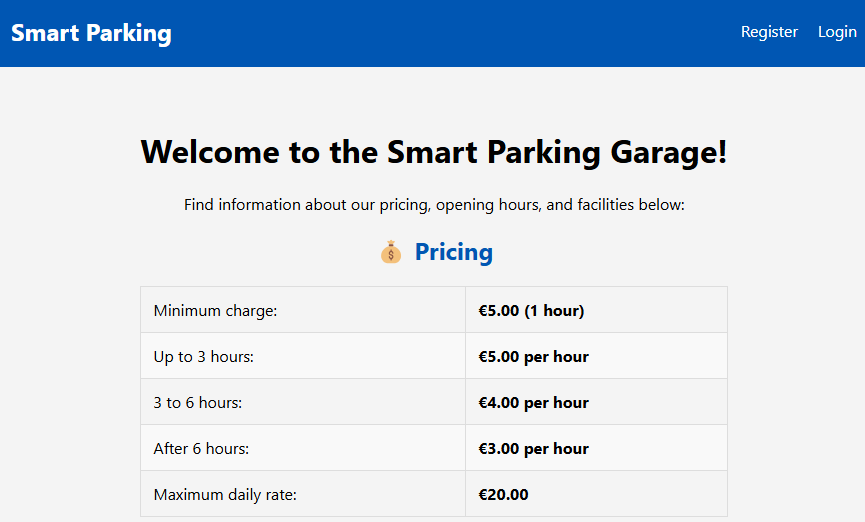
In mijn applicatie heb ik zorgvuldig meldingen geïmplementeerd om de gebruiker altijd goed op de hoogte te houden van de status van zijn of haar acties. Dit voorkomt verwarring en zorgt ervoor dat gebruikers weten wanneer er iets niet volledig is afgerond of wanneer er een probleem is. Een goed voorbeeld hiervan is het proces van het maken van een reservering. Wanneer een gebruiker probeert een reservering te plaatsen zonder eerst een auto te selecteren, verschijnt er een duidelijke melding die aangeeft dat er een fout is bij de reservering, waardoor de gebruiker direct weet wat er ontbreekt en wat hij of zij moet doen om verder te gaan.

Om deze meldingen visueel aantrekkelijker en gebruiksvriendelijker te maken, heb ik gekozen voor het gebruik van SweetAlert2. Dit is een Python-bibliotheek die speciaal is ontworpen voor het creëren van mooie en informatieve pop-up meldingen. SweetAlert2 biedt veel flexibiliteit en een moderne uitstraling, wat bijdraagt aan de algehele gebruikerservaring. Het stelt me in staat om meldingen te presenteren in een stijl die zowel opvallend als intuïtief is, zonder afbreuk te doen aan de functionaliteit van de applicatie.

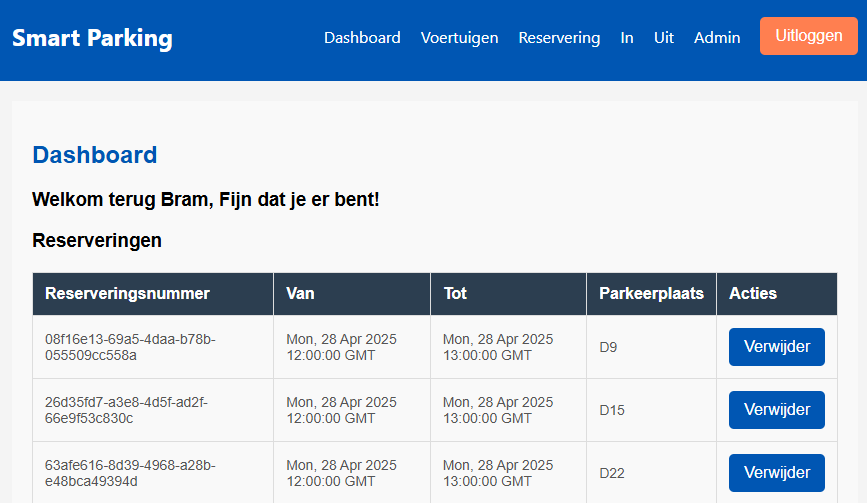
Daarnaast heb ik ervoor gezorgd dat de meldingen op verschillende punten in de applicatie verschijnen, zoals na het invullen van formulieren, bij het maken van reserveringen, en wanneer bepaalde acties worden afgerond. Dit zorgt voor een naadloze ervaring waarbij de gebruiker altijd weet wat er van hem of haar wordt verwacht.

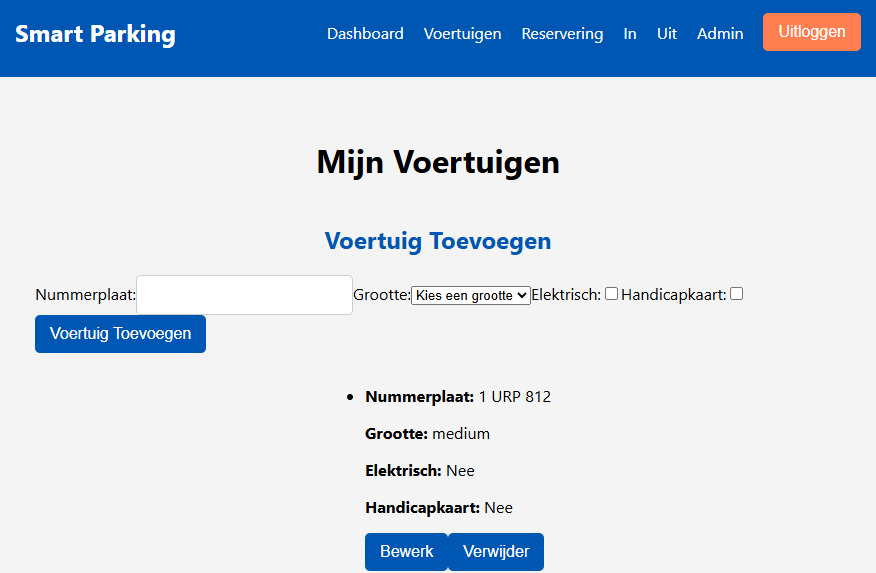
**9. Gebruikersflow**

**Stap 1**: Je begint op de **startpagina**, waar je de mogelijkheid hebt om een account aan te maken of in te loggen.

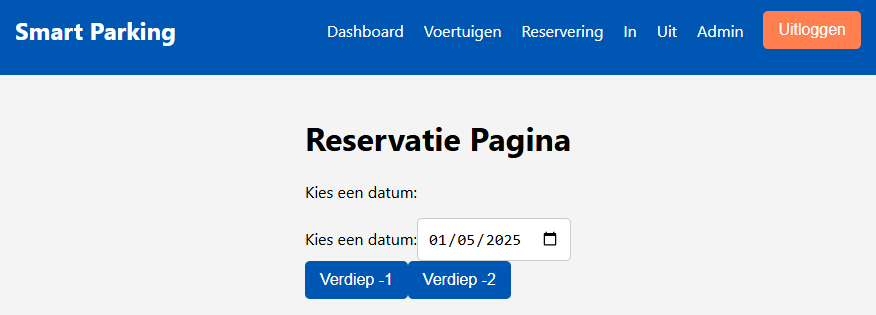


**Stap 2**: Na het inloggen kom je op het **dashboard**, waar je een overzicht van je al gemaakte reserveringen kunt zien.

  
**Stap 3**: Via het menu kun je naar **Voertuigen** gaan. Hier zie je een overzicht van je geregistreerde voertuigen en kun je nieuwe voertuigen toevoegen.



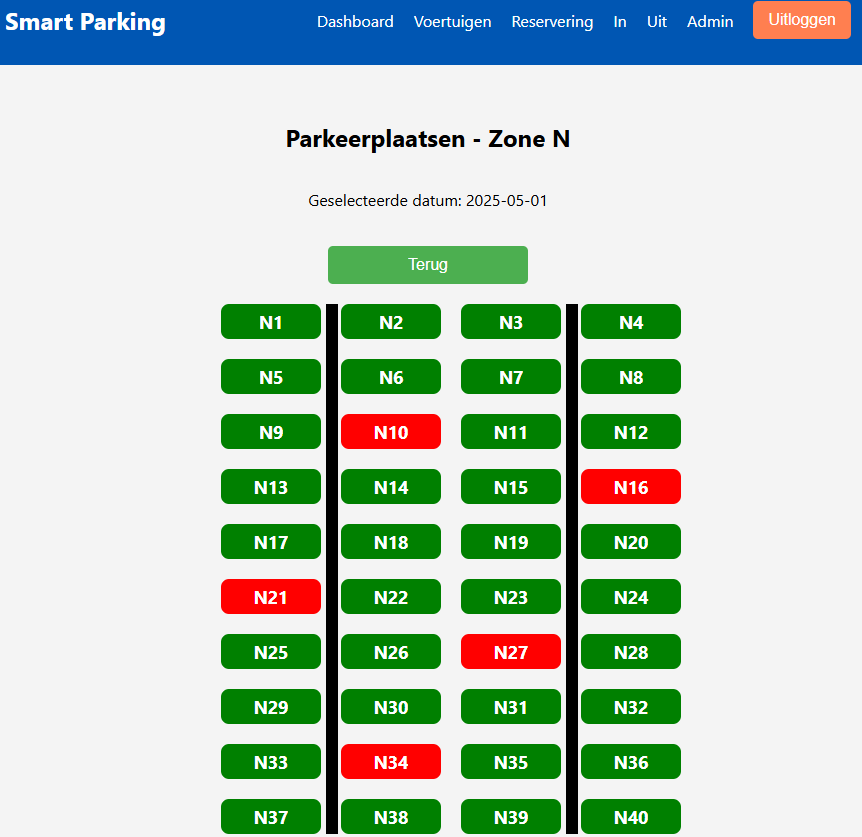
**Stap 4**: Zodra je voertuigen hebt geregistreerd, kun je een **reservering plaatsen**. Klik op de knop "Reservering" en vul de gevraagde datum in.

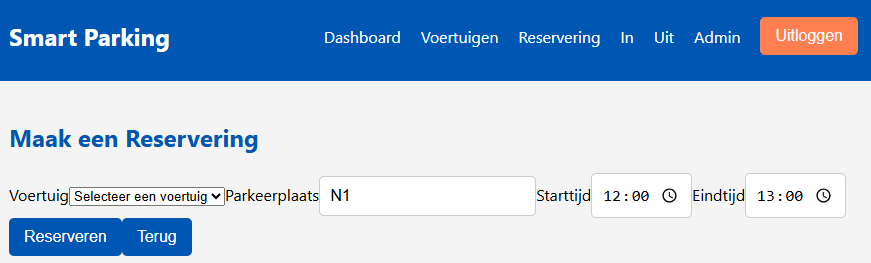
Vervolgens kies je een verdieping en een zone binnen de parkeergarage.



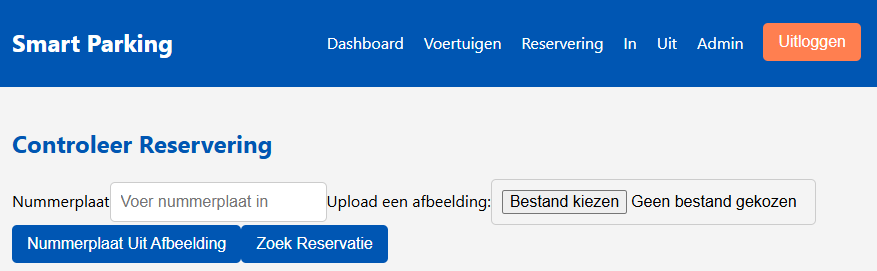
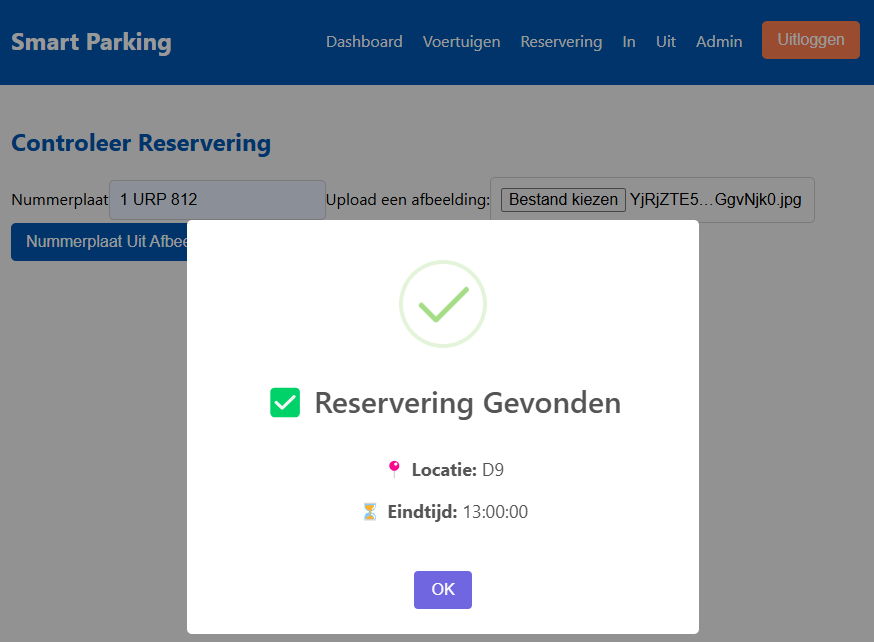
Verdieping

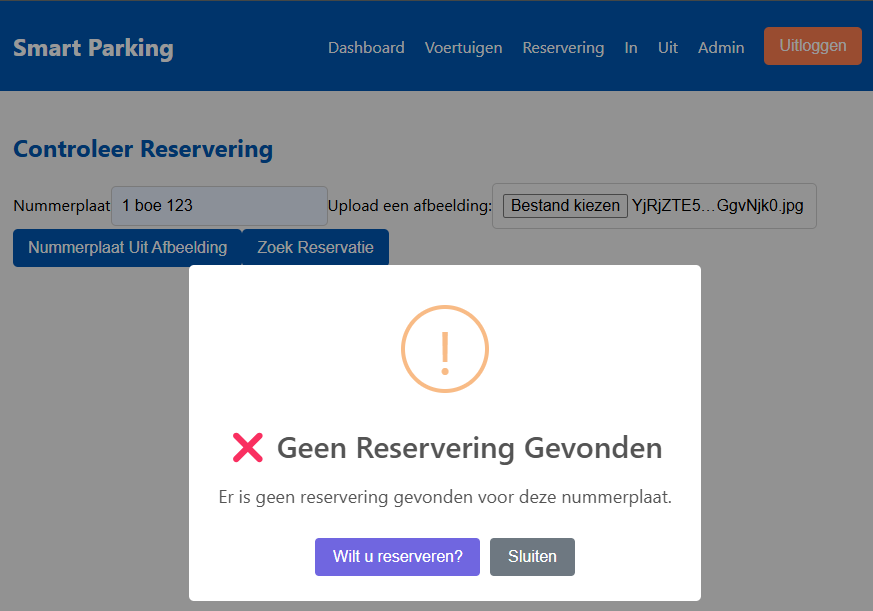
Als je dit hebt gedaan, selecteer je een **beschikbare groene parkeerplaats**,

vul je je gegevens in, en bevestig je de reservering.

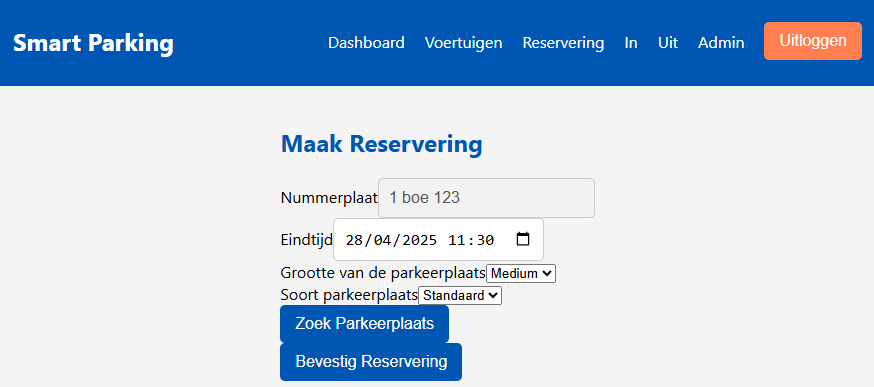


**Stap 5**: Wanneer je naar de parkeergarage gaat, moet je je **nummerplaat** of een afbeelding van je nummerplaat invoeren.

Het systeem zal vervolgens controleren of er een **geldig reservering** voor jouw nummerplaat bestaat. Als er een reservering is, krijg je de details van je reservering te zien. 

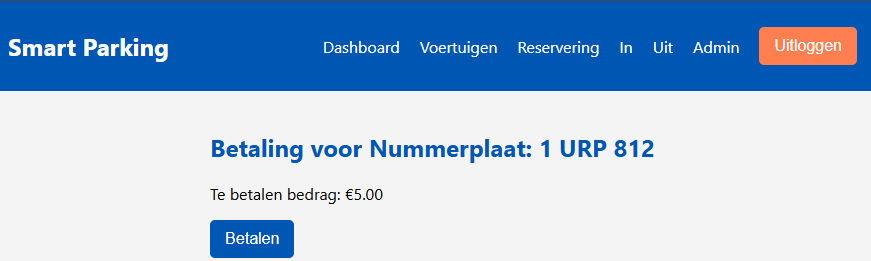
Als er geen reservering is, kun je snel een nieuwe reservering maken. Dit doe je door op de knop “Wilt u reserveren?” te klikken.  
****

En vervolgens de gevraagde gegevens in te vullen.



Vervolgens zoekt de applicatie een geschikte parkeerplaats, kun je de reservering bevestigen en mag je ook binnenrijden.

**Stap 6**: Bij het uitrijden geef je opnieuw je **nummerplaat** in.

Het systeem toont je dan het te betalen bedrag. Als alles klopt en je op "Betalen" klikt, wordt de betaling verwerkt en kun je uitrijden. 

**10. Problemen onderweg**

Tijdens het ontwikkelen van mijn applicatie ben ik verschillende problemen tegengekomen. Sommige daarvan hebben ervoor gezorgd dat ik moest afwijken van mijn oorspronkelijke plan.  
Een van de grootste uitdagingen was de nummerplaatherkenning. Mijn eerste idee was om dit te realiseren via een live videofeed: een camera zou continu filmen en de beelden direct naar de applicatie sturen, waar dan automatisch een herkenning van de nummerplaat zou gebeuren. Helaas bleek dit technisch te complex binnen de tijd die ik had en met de kennis waarover ik beschikte. Daarom heb ik gekozen voor een alternatief op basis van statische afbeeldingen. Deze aanpak was eenvoudiger te implementeren, maar vereist wel dat gebruikers zelf een foto uploaden.

Daarnaast wilde ik ervoor zorgen dat alle bevestigingsmails verzonden zouden worden via een echt, eigen e-mailadres. Ik had dit geprobeerd met een Google-account, maar liep vast omdat Google vereist dat je een gsm-nummer koppelt aan het account dat je hiervoor gebruikt. Mijn gsm-nummer was al gekoppeld aan mijn persoonlijke mail, waardoor dit niet mogelijk bleek. Ook kreeg ik tijdens het versturen van mails te maken met verbindingsproblemen. Na veel debuggen ontdekte ik dat het schoolnetwerk bepaalde poorten blokkeert, waardoor de mails niet verstuurd konden worden. Mijn oplossing hiervoor was om mijn smartphone als hotspot te gebruiken en op 4G te werken telkens wanneer ik de mailfunctionaliteit testte. Een eenvoudige, maar doeltreffende oplossing.

Daarnaast ben ik ook verschillende kleinere bugs tegengekomen. Soms werkten ID’s niet zoals verwacht of gaf de database foute informatie terug. Deze problemen kon ik meestal relatief snel oplossen door gebruik te maken van print()-statements. Zo kon ik stap voor stap nagaan waar het fout liep en wat de oorzaak was.



Een ander technisch probleem was dat mijn lokale server via XAMPP opeens niet meer opstartte. Dit gebeurde van de ene dag op de andere, zonder duidelijke reden. Na veel zoeken online heb ik besloten om XAMPP volledig opnieuw te installeren. Gelukkig werkte alles daarna weer zoals het moest.

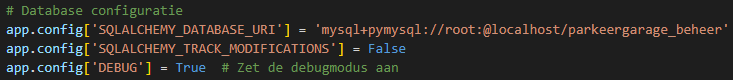
Hoewel het niet altijd makkelijk was, heb ik elk probleem kunnen oplossen, soms door logisch redeneren en testen, en soms door hulp te vragen of via ChatGPT mogelijke oplossingen op te zoeken.

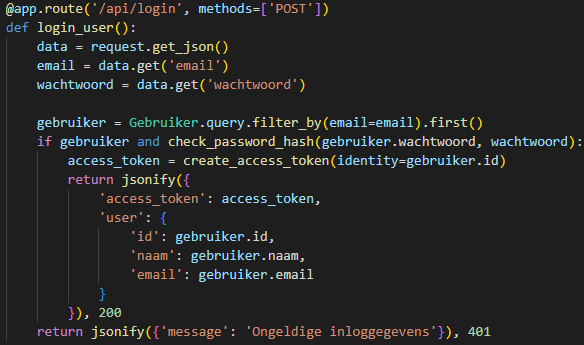
**11. Code**

Mijn applicatie bestaat uit verschillende codecomponenten die samenwerken om de functionaliteit van het systeem te bieden. De centrale backend van de applicatie is geschreven in **Flask (Python)**, waar alle functies en communicatie met de database plaatsvinden. Daarnaast heb ik voor elk apart tabblad een eigen **JavaScript**-bestand, zodat de structuur overzichtelijk en makkelijk te onderhouden is.

**Backend: Flask en SQLAlchemy**

De belangrijkste code voor de werking van de applicatie bevindt zich in de **Flask-backend**, die de centrale logica van de applicatie bevat. De databaseconfiguratie is essentieel voor het functioneren van de applicatie, aangezien deze de verbinding met de database tot stand brengt en de gegevens beheert. Voor de database maak ik gebruik van **SQLAlchemy**, omdat dit een krachtige en betrouwbare ORM is waarmee ik efficiënt kan werken met relationele databases. Ik heb er ook eerder mee gewerkt, waardoor ik er veel vertrouwen in heb.

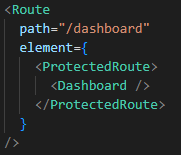
Hierin worden de configuraties voor de database verbinding gedefinieerd. Zonder deze configuratie zou de rest van de applicatie niet goed functioneren.

**Authenticatie: Inloggen en JWT Tokens**

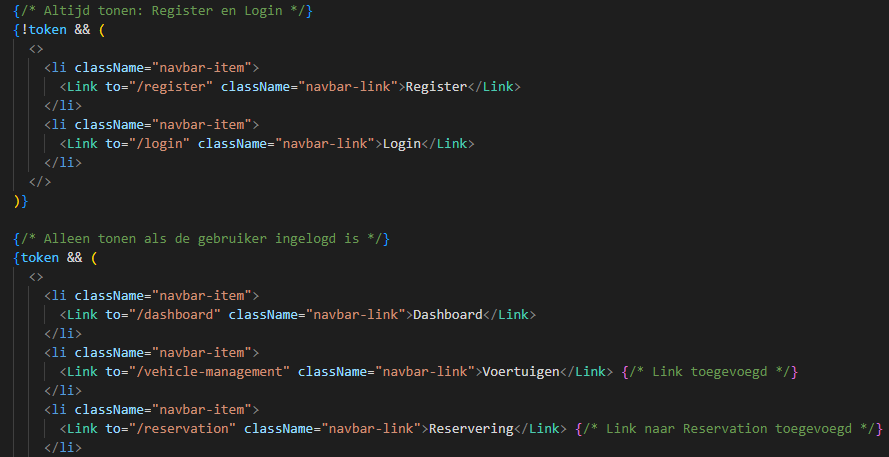
Een ander belangrijk onderdeel van de backend is de **inlogfunctionaliteit**. De inlogfunctionaliteit is geïmplementeerd als een **POST API-methode**, aangezien deze gegevens naar de database verzendt. Bij succesvolle authenticatie genereert de applicatie een **JWT-token** (JSON Web Token), waarmee wordt gecontroleerd of een gebruiker is ingelogd en toegang heeft tot beveiligde delen van de applicatie.

In dit gedeelte van de code wordt de JWT-token gegenereerd en opgeslagen, zodat de gebruiker na het inloggen toegang krijgt tot beveiligde pagina's.

**Frontend: JavaScript Bestanden**

Naast de backend bevat de applicatie ook verschillende **JavaScript-bestanden**. De belangrijkste hiervan is **app.js**, waarin alle routes worden gedefinieerd. Deze routes zijn essentieel voor de werking van de applicatie, omdat ze zorgen voor de navigatie tussen de verschillende tabbladen. Zonder deze routes zouden de links naar andere pagina's niet werken en zou de gebruiker niet kunnen navigeren.

Een ander belangrijk bestand is **Header.js**, dat samenwerkt met **app.js** en de navigatiebalk van de applicatie definieert. In dit bestand bepaal ik welke links zichtbaar zijn en waar ze naartoe leiden. Dit bestand bevat zowel **beveiligde** als **niet-beveiligde links**. Beveiligde links zijn alleen toegankelijk voor gebruikers die een geldige JWT-token hebben, wat voorkomt dat onbevoegde gebruikers toegang krijgen tot bepaalde delen van de applicatie.



Niet beveiligd

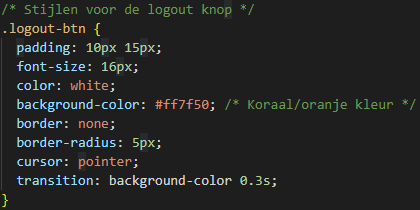
Beveiligd

**Gebruik van Axios voor API-aanroepen**

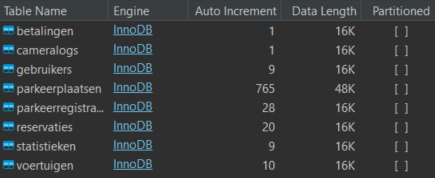
****In veel **JavaScript-bestanden** zie je de lijn import axios staan.

**Axios** is een JavaScript-gebaseerde bibliotheek die wordt gebruikt om HTTP-aanvragen te doen vanuit de browser. Het biedt een flexibele manier om **API-aanroepen** te maken, zowel voor het ophalen van gegevens als voor het verzenden van gegevens naar de backend.

**Stijlen en CSS**

Naast de functionaliteit van de applicatie is de **gebruikersinterface** ook belangrijk voor de algehele ervaring. Hiervoor gebruik ik verschillende **CSS-bestanden** om ervoor te zorgen dat de applicatie er consistent en aantrekkelijk uitziet. Deze bestanden bevatten de stijlen voor de verschillende elementen van de applicatie, zoals knoppen, formulieren en tekstvelden. Zonder deze CSS-bestanden zou de applicatie visueel minder aantrekkelijk zijn en zou de gebruikerservaring niet optimaal zijn.

**Database**

Mijn database bestaat uit acht verschillende tabellen, elk met een specifiek doel. Sommige tabellen bevatten enkel informatie, terwijl andere gebruikt worden om gegevens tijdelijk of permanent op te slaan. Op die manier werken alle onderdelen efficiënt samen en zorgen ze voor een vlotte werking van de applicatie. In het begin had ik nog enkele extra tabellen aangemaakt, maar tijdens de ontwikkeling bleek dat deze overbodig waren en zijn ze uiteindelijk verwijderd.

**Conclusie**

Dit is slechts een overzicht van de belangrijkste codecomponenten die mijn applicatie aandrijven. In werkelijkheid bevat het project veel meer code die verschillende functionaliteiten ondersteunt, zoals foutafhandeling, validatie van gebruikersinvoer, en het beheer van de interactie met de database. Al deze componenten werken samen om een functionele en gebruiksvriendelijke applicatie te bieden.

**12. Als ik meer tijd had...**

Mocht ik meer tijd gehad hebben, dan zou ik graag nog enkele verbeteringen en uitbreidingen aanbrengen aan mijn project.

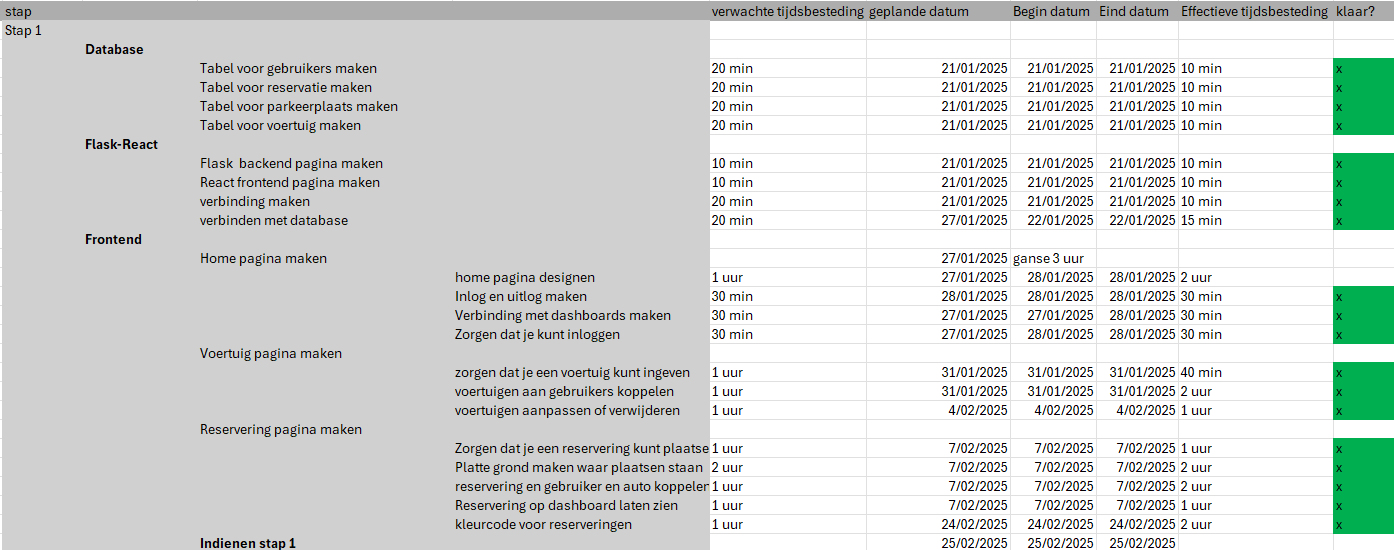
Zo zou ik de adminpagina nog uitgebreider en gebruiksvriendelijker maken. Denk hierbij aan grafieken die automatisch de bezetting per dag of week tonen, of tabellen met filters zodat je snel info over specifieke voertuigen of gebruikers kunt opzoeken. Ook een exportfunctie naar Excel of PDF zou handig zijn voor administratie.

Daarnaast zou ik de nummerplaatherkenning verder optimaliseren. Momenteel herkent het systeem al een groot deel van de platen, maar soms zijn er nog foutjes, zeker bij foto's met slechte belichting of een ongebruikelijke hoek. Door meer te testen, extra beeldfilters toe te passen en eventueel gebruik te maken van machine learning, zou ik dit verder kunnen verbeteren.

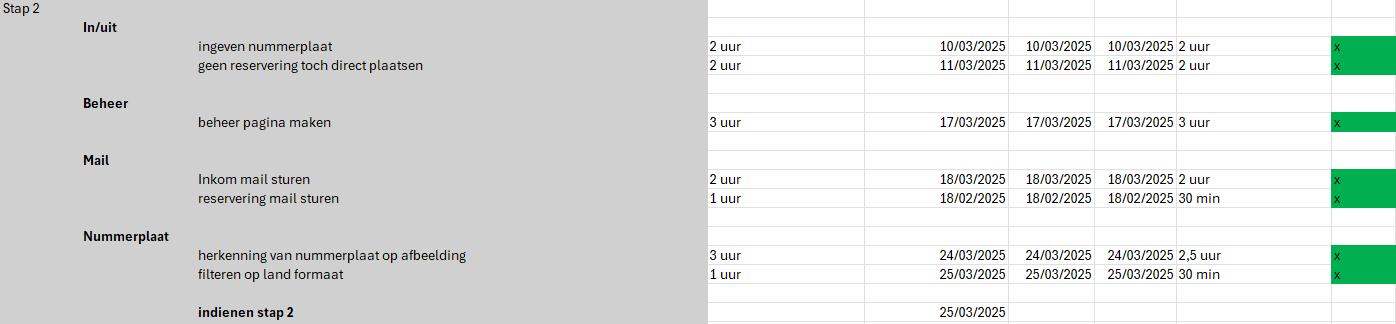
Ook zou ik mijn systeem willen koppelen aan een echte betaalprovider, zoals Mollie of Stripe, zodat de betalingen niet meer gesimuleerd hoeven te worden. Op die manier zou het project echt volledig functioneel kunnen zijn in een echte parkeergarage.

Ten slotte zou ik ervoor zorgen dat het hele systeem nog efficiënter draait. Hoewel het nu al stabiel is, zijn er altijd kleine optimalisaties mogelijk op vlak van snelheid, databasequeries en gebruikservaring.

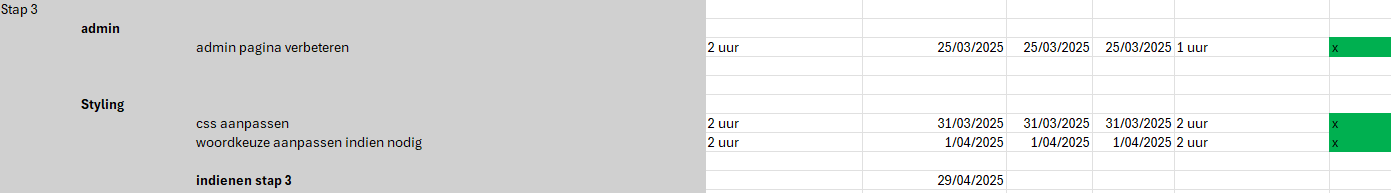
**13. timingtabel**



In stap 1 ging alles heel vlot, dit is zie je dan ook aan de timingtabel, er waren veel stappen maar telkens kleine dingen die niet extreem lang duurde. De meeste dingen gingen dan ook sneller dan dat ik ingeschat had. Het ging heel vlot, omdat dit stukjes zijn die ik vaak moet coderen en dus al weet hoe en wat alles moet en werkt.



In stap 2 zie je dan weer dat er veel minder delen zijn, en dat deze langer duren. Dit komt omdat alles wat moeilijker begon te worden, en omdat er hier wat meer error codes kwamen.



Stap 3 heeft dan nog minder delen, dit omdat het zo goed als af was en er dus ook niet meer heel veel moest gebeuren. Dit waren dan ook enkel nog de laatste paar kleine dingen die moesten gebeuren zodat de applicatie volledig is.

Besluit

Het ontwikkelen van de slimme parkeergarage was een uitdagend maar leerzaam proces. Tijdens het project kwam ik verschillende technische obstakels tegen, zoals het implementeren van de nummerplaatherkenning en het versturen van e-mails via een extern systeem. Toch heb ik elke uitdaging weten te overwinnen door logisch redeneren, het testen van verschillende oplossingen en soms door hulp in te roepen van externe bronnen zoals forums en ChatGPT. De toepassing is niet perfect, maar het biedt wel de basis voor een systeem dat met enkele aanpassingen in een echte parkeergarage zou kunnen worden ingezet.

Persoonlijk ben ik tevreden met het eindresultaat, vooral gezien de beperkte tijd en middelen. Als ik terugkijk, zou ik echter enkele dingen anders aanpakken. Zo zou ik meer tijd hebben besteed aan het optimaliseren van de nummerplaatherkenning door gebruik te maken van geavanceerdere technieken of machine learning. Ook had ik het reserveringssysteem wellicht kunnen uitbreiden met meer gedetailleerde functies, zoals dynamische prijsstelling op basis van de vraag. Als ik opnieuw aan een dergelijk project zou beginnen, zou ik me waarschijnlijk meer richten op het verbeteren van de betrouwbaarheid van bepaalde functionaliteiten, zoals de nummerplaatherkenning, en het optimaliseren van de gebruikerservaring.

De GIP heeft me niet alleen geholpen om mijn technische vaardigheden te verbeteren, maar ook om mijn probleemoplossend vermogen en mijn vermogen om te werken met verschillende technologieën te versterken. Uiteindelijk is dit project een goed voorbeeld van hoe een concept, zelfs met beperkingen, kan leiden tot een functionele en nuttige toepassing.

Bibliografie

Bijlagen