Project Individueel

Ontwikkeling van een Factuurprogramma voor Nieuwbouwprojecten:

Jens Van de Velde
19/12/2023

Index

Titel	1
Inleiding	3
Technologiebeschrijving en Architectuur	3
Opdrachtsbeschrijving en Functionele Eisen	3
Ontwikkelingsstrategie	4
Resultaten en Praktische Toepassingen	4
Conclussies en Reflectie	4
Referenties	5
Opbouw van de structuur	6
De Kern van het Factuurprogramma	7
Python Backend Architectuur	9
De Frontend van het Factuurprogramma	11
Project opstarten	13

Factuurprogramma voor Nieuw Bouwprojecten

Inleiding

Ik ben enthousiast om een programma te delen dat ik heb gemaakt om het creeren en het opvolgen van facturen in de bouw veel makkelijker te maken. Voor mijn project heb ik hard gewerkt aan een gebruiksvriendelijk factuurprogramma. Dit programma is er gekomen omdat er vroeger veel tijd gespendeerd werd met klantgegevens in Excel te plaatsen en elke factuur met de hand in Excel aan te maken. Mijn nieuwe programma maakt alles veel sneller en minder foutgevoelig, zodat mijn klant zich kan focussen op het bouwen zelf en hun klanten beter kunnen helpen.

Technologiebeschrijving en Architectuur

Ik heb een factuurprogramma opgezet met coole nieuwe technologie en slimme manieren van programmeren. Het hart van het programma is Python, een krachtige taal, samen met React voor het deel dat gebruikers zien. Ik heb ook HTML en CSS gebruikt om het er goed uit te laten zien. Dit alles maakt het makkelijk en fijn voor gebruikers om met het programma te werken. Met Python's CORS API kan het programma veilig gegevens delen over het internet, wat echt belangrijk is tegenwoordig. Voor de database gebruiken we Azure SQL Edge op een Raspberry Pi, wat betekent dat we gegevens zowel lokaal als in de cloud kunnen beheren. Al deze technologieën heb ik bij elkaar gebracht om een programma te maken dat makkelijk, snel en betrouwbaar werkt.

Opdrachtbeschrijving en Functionele Eisen

Het doel van mijn project was om een factuurprogramma te maken dat perfect past bij wat de klant nodig heeft voor nieuwe bouwprojecten. Elke factuur moet uniek gemaakt worden, met gegevens rechtstreeks uit de database. Het programma moet ook kunnen omgaan met ingewikkelde facturen, automatisch dingen berekenen op basis van de prijzen die werden afgesproken, en alles goed op de factuur in te vullen. Het moet ook makkelijk zijn om klantgegevens en oude facturen te vinden en op te vragen, alsook moet de klant gegevens uit Excel-bestanden kunnen importeren in het programma.

Ontwikkelingsstrategie

Ik ben gedoken in de programmeerwereld om dit te kunnen verwezelijken. Ik heb mezelf Python en React aangeleerd en dat samen met mijn lessen op school was de perfecte basis om dit project te starten. Met mijn nieuwe kennis heb ik een eerste versie van het programma gemaakt en deze steeds verbeterd door te luisteren naar wat de mede studenten en familie zeiden. Het was belangrijk voor me dat het programma makkelijk en precies werkt, zodat het factureren soepel gaat, daarnaast heb ik online ook heel veel zaken opgezocht en geleerd tijdens het programmeren. Vaak als ik op een probleem stuite heb ik bepaalde zaken moeten opzoeken.

Resultaten en Praktische Toepassingen

Het eindresultaat is een factuurprogramma dat past bij wat de klant nodig heeft in de bouw. Het maakt het hele proces van factureren een stuk makkelijker, en het zorgt ervoor dat de facturen kloppen en er professioneel uitzien. De klant kan ook makkelijk gegevens importeren, en dat heb ik getest en het werkt goed. Echter is dit nog maar de eerst werkende versie en moet er nog meer uitgebreid worden aan het programma. Dit is echter voor de volgende stadiums.

Conclusies en Reflectie

Dit project heeft me laten zien hoe belangrijk het is om software te maken die echt past bij wat mensen nodig hebben. Ik heb veel geleerd over programmeren en over hoe je een programma maakt dat niet alleen goed werkt maar ook echt helpt bij het dagelijks werk. Echter heb ik wel beseft dat niet alles heel soepel loopt en dat er vaak heel veel momenten zijn dat je online zaken moet gaan bijleren of opzoeken om verder te kunnen werken in het programma.

Referenties

Bibliotheken en Modules

1. **PDFKit**: Deze bibliotheek wordt gebruikt voor het omzetten van HTML naar PDF. Het is een essentiële tool binnen ons programma voor het genereren van PDF-documenten uit webinhoud of HTML-templates.
2. **datetime (from datetime import datetime):** Deze module is cruciaal voor het hanteren van datums en tijden binnen het programma. Het stelt ons in staat om de huidige tijd te verkrijgen, te manipuleren en op een flexibele manier te formatteren.
3. **locale:** Deze module helpt bij het internationaliseren van deze applicatie. Het maakt het mogelijk om lokale instellingen zoals datum- en tijdformaten en valutatekens aan te passen, wat van groot belang is voor de veelzijdigheid van het programma.
4. **os:** Deze module biedt interactie met het besturingssysteem. We gebruiken het voor bestands- en directorybeheer, evenals voor het verkrijgen van omgevingsvariabelen, een sleutelfunctionaliteit voor ons programma.
5. **pyodbc:** Deze module wordt gebruikt voor databaseconnectiviteit via ODBC. Het is essentieel voor het lezen van en schrijven naar diverse databasetypes vanuit ons programma.
6. **Flask:** Dit lichtgewicht webapplicatieframework is de ruggengraat van ons programma voor het bouwen van webapplicaties. Het biedt een eenvoudige en begrijpelijke syntax voor de ontwikkeling van onze applicatie.
7. **Flask-RESTx**: Als uitbreiding op Flask, vereenvoudigt Flask-RESTx het proces van het bouwen van REST APIs. Het biedt nuttige hulpmiddelen zoals resource routing en input/output parsing, die we in ons programma gebruiken.
8. **CORS (Cross-Origin Resource Sharing):** Door gebruik te maken van de Flask-CORS uitbreiding, kunnen we in ons programma veilig beheren hoe bronnen in webpagina's worden geladen vanuit verschillende domeinen, een belangrijk aspect voor de beveiliging en functionaliteit van onze webapplicatie.
9. **subprocess:** Deze module stelt ons in staat om nieuwe processen te starten en met deze processen te communiceren. Het is een integraal onderdeel van ons programma, vooral voor taken die externe processen vereisen.
10. **Dropbox:** Met deze module kunnen we interageren met de Dropbox API, wat ons in staat stelt om bestanden te beheren in een Dropbox-account. Dit is cruciaal voor het bestandsbeheer binnen ons programma.

Opbouw van de structuur

Python Back-end Architectuur

Voor de motor van mijn programma heb ik me verdiept in Python. Dit is omdat Python krachtig genoeg is en mij toelaat om complexe taken op een eenvoudige manier uit te voeren. Ik heb een reeks speciale klassen gecreëerd die elk hun eigen taak hebben, wat mij helpt het overzicht te bewaren en efficiëntie te verzekeren.

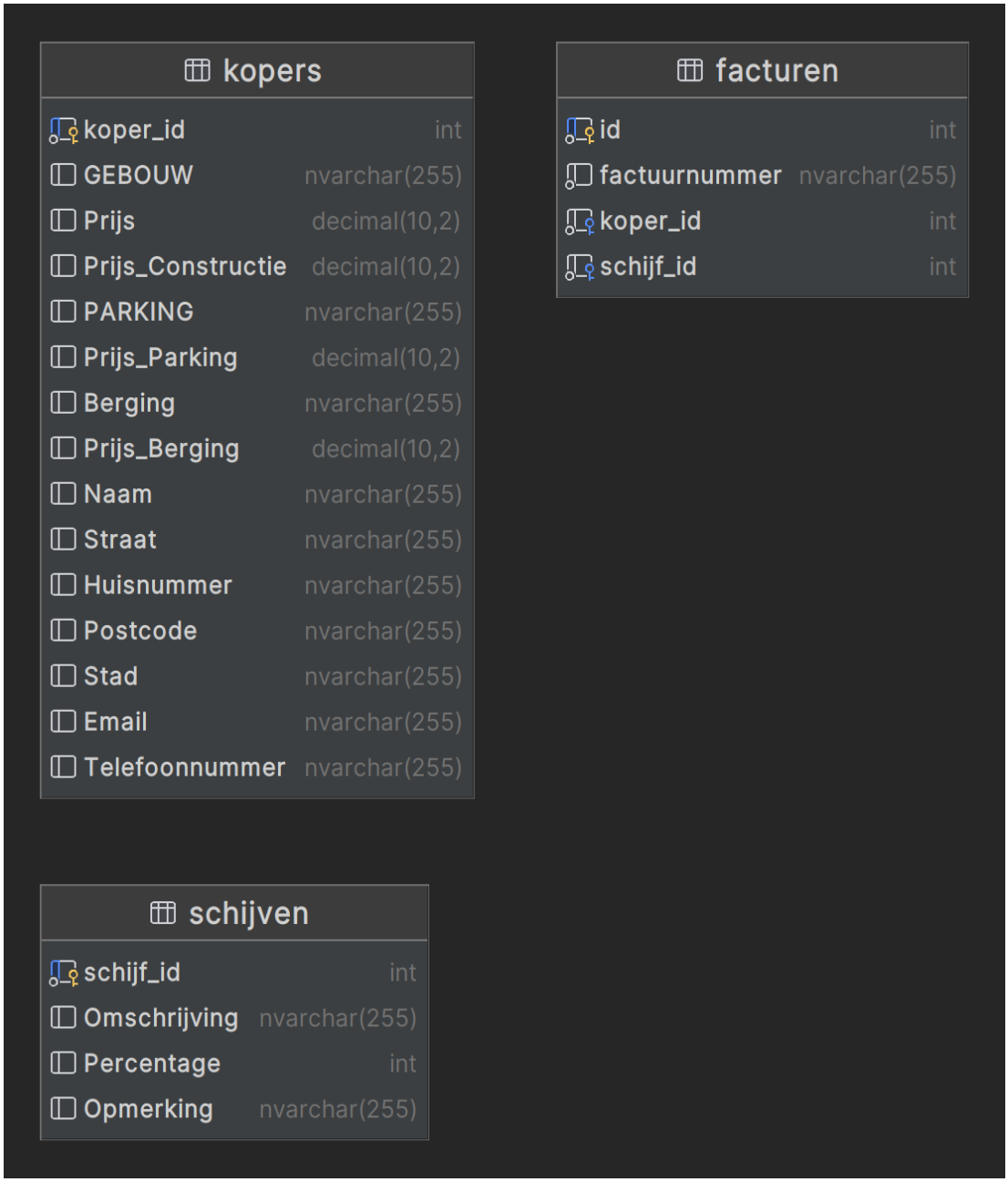
React en Front-end Design Filosofie

Aan de voorkant heb ik React in de mix gegooid. Waarom? Omdat het soepel is en mij helpt een strakke, interactieve gebruikersinterface te bouwen. Het maakt het dagelijkse gebruik een stuk aangenamer en zorgt ervoor dat het programma er professioneel uitziet en aanvoelt.

CORS Implementatie en API-beveiliging

Voor de veiligheid en flexibiliteit hebben ik CORS geïntegreerd in onze Python API. Dit is van onschatbare waarde omdat het mij de mogelijkheid geeft om veilig gegevens te delen over verschillende bronnen heen, wat een must is in een wereld waarin applicaties constant 'praten' met elkaar.

De Kern van het Factuurprogramma



Ontwerpfilosofie en Structuur

Bij het ontwerpen van de database was mijn doel helderheid en stevigheid, daarom heb ik het beperkt tot slechts drie essentiële tabellen: Kopers, Facturen en Schijven. Elk speelt een sleutelrol en zorgt ervoor dat het systeem soepel draait. 'Kopers' bewaart klantgegevens, 'Facturen' houdt elke transactie bij en 'Schijven' deelt de betalingen op in beheersbare delen. Dit zorgt voor een helder overzicht en eenvoudige toegang tot alle informatie.

Strategische Voordelen

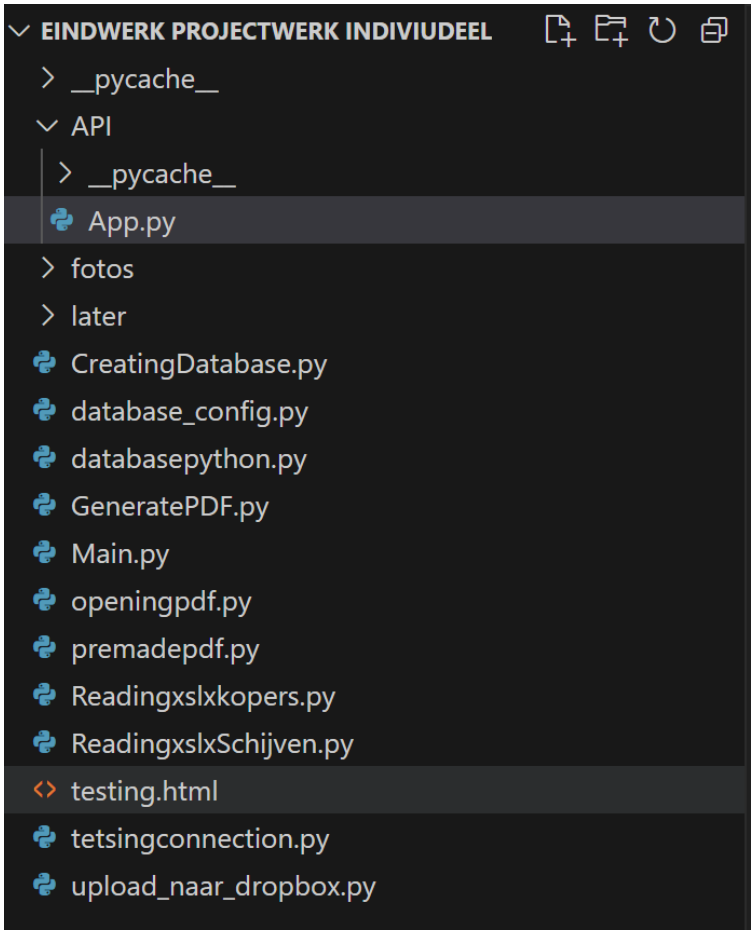
Deze gestroomlijnde structuur maakt navigatie door het systeem intuïtief en efficiënt. Doordat ik het simpel hou, is de kans op fouten kleiner en kan ik sneller werken. Dit minimalisme betekent dat de klant zich beter kan richten op het verbeteren van de bouwervaring voor hun klanten, zonder hun zorgen te maken over administratieve rompslomp.

Toekomstbestendigheid en Schaalbaarheid

Bij het opzetten van de database ging het mij om een solide basis die op lange termijn standhoudt. Het is zo gebouwd dat het robuust en betrouwbaar is voor de huidige functies. Voor nu is het systeem specifiek afgestemd op de klant hun huidige processen, wat betekent dat het niet onmiddellijk is ontworpen met uitbreidbaarheid in gedachten. Dit komt omdat de klant een gespecialiseerde oplossing nodig had die precies doet wat hun vereisen waren.

Als de klant in de toekomst meer functies nodig heeft, kan ik bekijken hoe ik het systeem kan aanpassen of uitbreiden. Dit kan betekenen dat ik bepaalde delen moeten herzien of opnieuw moeten opbouwen om aan nieuwe eisen te voldoen. Maar dat is een brug waar we overheen gaan als we er aankomen. Voor nu is het belangrijk dat het systeem doet wat het moet doen en dat het goed doet.

Python Backend Architectuur: De Kracht Achter de Schermen



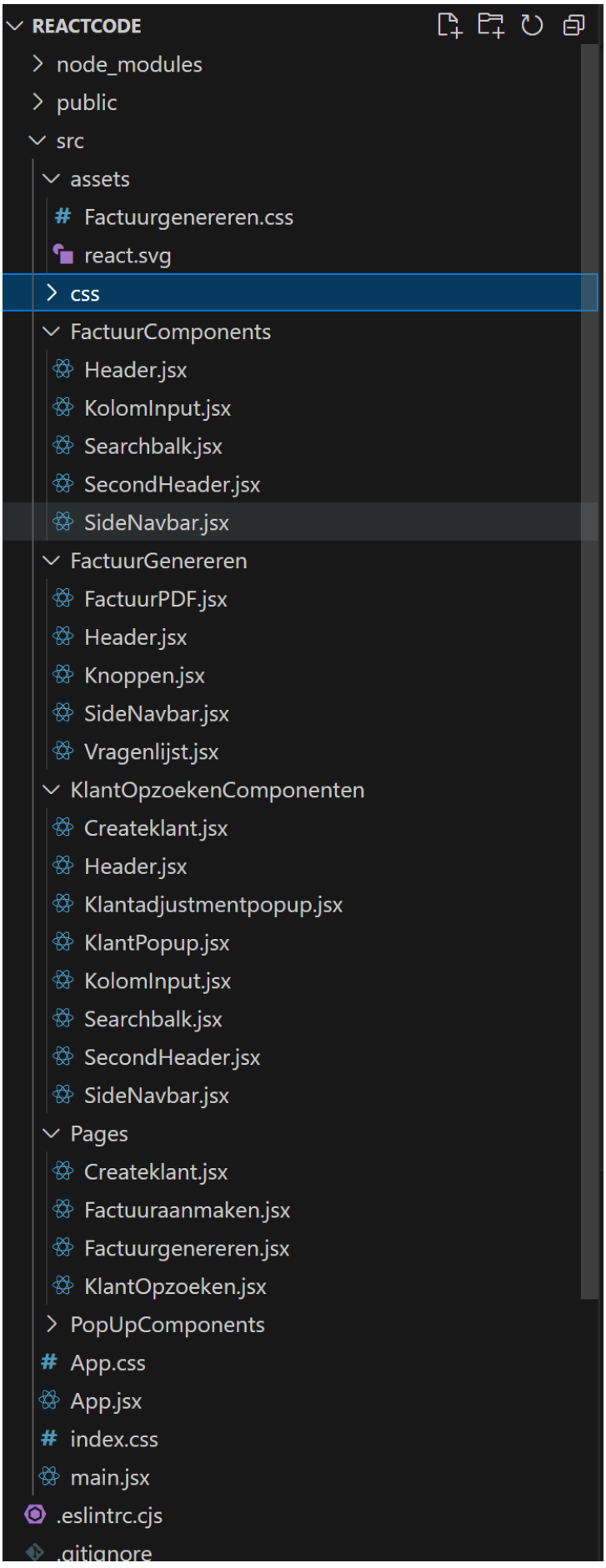
Voor de achterkant van het facturatiesysteem heb ik Python gebruikt, vanwege zijn veelzijdigheid en krachtige bibliotheken. Ik heb specifieke scripts ontwikkeld voor elke taak:

- CreatingDatabase.py zet de database op, zorgt ervoor dat de juiste tabellen aangemaakt worden.
- database_config.py en databasepython.py voorziet de connectionstring van onze online database.
- Met GeneratePDF.py zetten we gegevens om in professioneel ogende PDF-facturen.
- Main.py is het hart van de operatie, waar alles samenkomt en wordt gecoördineerd.
- openingpdf.py is speciaal voor het beheren van PDF-bestanden.

- Voor het importeren van gegevens hebben we Readingxlsxkopers.py en ReadingxlsxSchijven.py, die de Excel-bestanden met klant- en betalingsinformatie inlezen.
- En voor het testen van de verbindingen binnen het systeem is er testingconnection.py.

Alles bij elkaar zorgt deze opzet voor een soepele, efficiënte en betrouwbare backend die het facturatiesysteem aandrijft. En net als bij de front-end, is het belangrijk dat alles duidelijk en onderhoudsvriendelijk is. Mocht er in de toekomst iets veranderen of moet ik iets toevoegen, dan is dat met deze opzet geen probleem.

De Frontend van het Factuurprogramma



Toen ik aan de slag ging met het bouwen van de voorkant voor het facturatieprogramma, heb ik voor React gekozen. Waarom? Nou, React is gewoon super handig voor het maken van onderdelen die we steeds weer kunnen gebruiken. Ik heb het zo opgezet dat elk stukje, van de lijst met facturen tot de info van een klant, zijn eigen plekje heeft. Dat maakt het een stuk overzichtelijker en makkelijker om dingen later aan te passen.

Ik heb echt mijn best gedaan om het simpel te houden voor iedereen die het gebruikt. Het moet gewoon logisch zijn hoe je alles doet in de app, zonder dat je erbij hoeft na te denken. En het moet snel werken, dus ik heb ervoor gezorgd dat React niet meer doet dan nodig is – dat houdt het allemaal vlot.

Voor de stijl heb ik iets cools gebruikt: CSS-modules. Die zorgen ervoor dat alles er strak uitziet, op welk apparaat je ook werkt. En als er eens iets fout gaat, geen paniek: ik heb ervoor gezorgd dat de app je vertelt wat er mis is en wat je moet doen. Zo kan iedereen makkelijk z'n weg vinden in de app, en als er iets nieuws moet gebeuren, dan is dat geen big deal.

Project opstarten

Om het project uit te pakken en aan de slag te gaan, ga je als volgt te werk:

Wanneer je het ZIP-bestand uitpakt, zie je vier mappen. Begin met de makkelijke 'Facturen'-map – hier zit 1 voorbeeldfactuur in en is bedoeld om facturen op te slaan die je later gaat aanmaken.

In de 'pythonfile'-map vind je een Excel-bestand met belangrijke opstartdata. Deze data wordt gebruikt om de database mee te vullen.

Laten we het project opstarten met de twee programma's: het Python-programma en het React-programma. Open beide mappen, 'PythonCode' en 'ReactCode', in Visual Studio Code.

Als je de database op je eigen systeem wilt draaien, pas dan de instellingen in het 'database_config.py'-bestand aan naar jouw setup. Het is allemaal ingesteld voor gebruik met een SQL Server-database.

Na eventuele aanpassingen start je het programma op door in de terminal 'python main.py' te tikken en op Enter te drukken. Dit test de databaseverbinding, zet de database klaar en pompt vervolgens alle gegevens uit het Excel-bestand de database in.

Om het Python-deel te starten, typ je 'cd API' in de terminal, druk je op Enter en dan start je het met 'python app.py'.

Voor de React-app open je een ander venster in Visual Studio Code. Daar is het proces straightforward: open de terminal, voer 'yarn install' uit om alles te installeren, en met 'yarn dev' start je het visuele op en komt dit tot leven. Surf naar je localhost om te zien wat je hebt gebouwd.