Geautomatiseerde prijsvergelijking voor Belgische supermarktketens.

Een Proof of Concept-systeem voor de in Gent aanwezige supermarktketens, gebaseerd op openbaar toegankelijke data.

Aliaksandra Nemchynava.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Mevr. L. Vuyge
Co-promotor: Dhr. J. Pots
Academiejaar: 2024–2025
Eerste examenperiode

Departement IT en Digitale Innovatie .

HO
GENT

Woord vooraf

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Samenvatting

De stijgende voedselprijzen in België leggen een groeiende financiële druk op studenten met een beperkt budget en beperkte mobiliteit. Bestaande prijsvergelijkingstools houden onvoldoende rekening met praktische beperkingen van winkelen, zoals reisafstand en het aantal winkels dat een consument bereid is te bezoeken. Bovendien baseren veel tools zich op barcodevergelijking, waardoor huismerken vaak buiten de scope worden gelaten. Daarnaast staan veel van deze diensten of hun extra functies achter een betaalmuur.

Dit onderzoek richt zich op het ontwerpen van een systeem dat consumenten, met name studenten in Gent, ondersteunt om weloverwogen beslissingen te kunnen nemen over hun boodschappen. De centrale onderzoeksvraag luidt: “Hoe kan een transparant en schaalbaar systeem worden ontworpen en ontwikkeld om automatisch supermarktprijzen in België te verzamelen, matchen en vergelijken, rekening houdend met beperkingen, zoals afstand en het aantal te bezoeken winkels?”

Het voorgestelde prototype wordt ontwikkeld in Python en Django, waarbij gebruik wordt gemaakt van webscrapingstechnieken om prijsgegevens van Belgische supermarktwebsites te verzamelen. De gegevens worden opgeslagen in een PostgreSQL-database. Gebruikers kunnen een boodschappenlijst invoeren en een maximale reisafstand of een limiet voor het aantal winkels instellen. Vervolgens wordt het meest kostefficiënte aankoopplan door het systeem berekend. De gegenereerde voorstellen worden geëvalueerd door hun besparingen te vergelijken met een aankoop van alle boodschappen in één winkel, op basis van vooraf gedefinieerde boodschappenlijsten. De resultaten zijn merkbare besparingen over alle voorgestelde boodschappenlijsten.

Dit project draagt bij aan een realistisch en toegankelijk prijsvergelijkingssysteem gericht op studenten door technische efficiëntie te combineren met consumentgerichte beperkingen. Dit stelt studenten in staat bewuster en voordeliger te winkelen.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Lijst van codefragmenten

1

Inleiding

De voedselprijzen in België zijn de afgelopen jaren aanzienlijk gestegen, wat een groeiende financiële druk legt op studenten en andere budgetbewuste consumenten. Hoewel er verschillende prijsvergelijkingstools voor supermarkten bestaan, richten deze zich over het algemeen op het presenteren van de laagste prijzen zonder rekening te houden met praktische beperkingen, zoals de afstand die een consument bereid is af te leggen of het aantal winkels dat hij of zij redelijkerwijs kan bezoeken. Hierdoor bieden deze tools theoretisch optimale oplossingen die in de praktijk niet haalbaar zijn, vooral voor studenten met beperkte mobiliteit en een strak schema.

1.1. Probleemstelling

Studenten ondervinden vaak uitdagingen bij het vinden van de meest kostenefficiënte winkelopties. Beperkte budgetten, gecombineerd met tijd- en reisbeperkingen, bemoeilijken efficiënte prijsvergelijkingen tussen verschillende supermarkten. Bestaande tools houden zelden rekening met deze beperkingen, waardoor er een kloof ontstaat tussen beschikbare prijsinformatie en bruikbare, gebruikersgerichte inzichten. Dit onderzoek pakt deze kloof aan door zich te richten op de ontwikkeling van een systeem dat afgestemd is op de behoeften van studenten met een beperkt budget in Gent.

1.2. Onderzoeksvraag

Om een dergelijk systeem te ontwikkelen, moet de volgende hoofdonderzoeksvraag worden beantwoord: Hoe ontwerp en ontwikkel je een transparant en schaalbaar systeem dat automatisch supermarktprijzen in Gent kan verzamelen, matchen en vergelijken, rekening houdend met de afstand tot de consument en een door de gebruiker opgegeven limiet voor het aantal te bezoeken winkels?

1.3. Deelvragen

Verder moet onderzoek worden gedaan naar de programmatische en architecturale details van de beoogde oplossing en de succesfactoren ervan. Meer specifiek:

- Hoe kunnen supermarktprijsgegevens automatisch worden verzameld en gestructureerd?
- Welke benaderingen kunnen worden gebruikt om algemene productnamen te matchen en zo nauwkeurige vergelijkingen te maken?
- Hoe kan het systeem de meest kosteneffectieve combinaties van winkels binnen een door de gebruiker gedefinieerde afstand en een winkellimiet berekenen en aanbevelen?
- Hoe kan de systeemarchitectuur worden ontworpen om schaalbaarheid en transparantie van de data te ondersteunen?
- Aan welke criteria moet het prototype voldoen om als een geldig proof-of-concept te worden beschouwd?

Om deze vragen te beantwoorden en het onderzoek te sturen, is inzicht vereist in de doelgroep en de probleemcontext. Meer specifiek:

- Welke factoren beïnvloeden momenteel de consumptiegewoonten van studenten in België?
- Welke tools voor prijsvergelijking zijn er in België beschikbaar en welke tekortkomingen hebben ze voor studenten?
- Welke technische en praktische uitdagingen zijn er bij het verzamelen van prijsgegevens van Belgische supermarkten?

1.4. Onderzoeksdoelstelling

Het resultaat van dit onderzoek is een prototype, geïmplementeerd met Python en Django, dat prijsgegevens verzamelt via webscraping van Belgische supermarktwebsites. Gebruikers van dit prototype kunnen een algemene boodschappenlijst invoeren en een maximale reisafstand of een limiet voor het aantal winkels opgeven. Het systeem berekent vervolgens de meest kostenefficiënte combinatie van winkels op basis van deze beperkingen. Het systeem wordt geëvalueerd met behulp van een vooraf gedefinieerde standaardboodschappenlijst voor studenten om de totale kosten van een aankoop in één winkel te vergelijken met de door het systeem gegenereerde, geoptimaliseerde aanbeveling voor meerdere winkels.
Dit onderzoek draagt bij aan de ontwikkeling van realistische en toegankelijke tools voor supermarktprijsvergelijking die technische efficiëntie combineren met consumentgerichte beperkingen. Door zich te richten op de behoeften van studenten,

beoogt het systeem de prijstransparantie te vergroten en weloverwogen, budgetbewuste winkelbeslissingen te ondersteunen. Het biedt praktische inzichten die toekomstige consumentgerichte toepassingen kunnen inspireren.

1.5. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:
In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.
In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.
In Hoofdstuk 6, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

In het vorige hoofdstuk is de probleemstelling geschetst: studenten in Gent hebben behoefte aan een prijsvergelijkingssysteem dat rekening houdt met praktische beperkingen zoals reisafstand en een beperkt budget. Dit hoofdstuk plaatst dat probleem in een bredere context door de stand van zaken in het onderzoeksdomein te beschrijven. Digitale prijsvergelijkingstools en geautomatiseerde dataverzameling worden steeds belangrijker in de voedselretail. Dit onderzoek situeert zich op het kruispunt van drie relevante domeinen: webgebaseerde dataverzameling, productnormalisatie- en prijsvergelijkingssystemen voor consumenten.

2.1. Context: voedselprijzen en studentendruk

Recente economische indicatoren van België wijzen op aanhoudende prijsdruk op voeding. Volgens het CPI-rapport van Statbel blijven de algemene en kerninflatie gedurende 2024-2025 hoog, met een kerninflatie van meer dan 2% in oktober 2025(Statbel, 2025a, 2025b). Onafhankelijke tracking door Testaankoop/Testachats meldt eveneens een supermarktspecifieke inflatie van ongeveer 4% in 2025(Testaankoop, 2025a, 2025b). Bredere macro-economische analyses(*OECD Economic Surveys: Belgium 2024*, 2024) tonen de gedetailleerde impact van inflatie aan en bevestigen de prijsdruk op consumenten. Samen onderbouwen deze bronnen de relevantie van het probleem voor prijsgevoelige groepen zoals studenten.

2.2. Bestaande oplossingen

Er zijn verschillende specifieke Belgische tools beschikbaar om consumenten te helpen supermarktprijzen te vergelijken en betere producten te selecteren, zoals PingPrice (PingPrice, 2024) en G4U (G4U, 2025). Beide apps hebben echter hun beperkingen. PingPrice vergelijkt producten met behulp van barcodes, waardoor het geen effectieve vergelijking kan maken tussen huismerkproducten of gene-

rieke producten die geen gestandaardiseerde identificatiecodes hebben. Hierdoor worden veel relevante artikelen uitgesloten van vergelijkingen.
G4U biedt daarentegen uitgebreide product- en promotie-informatie, maar werkt als een betaalde dienst, waardoor de toegankelijkheid beperkt is voor studenten die al met financiële beperkingen kampen. Daarom is er behoefte aan een gratis en transparant alternatief waarmee gebruikers generieke productcategorieën kunnen vergelijken in plaats van barcodes.
In Tabel 2.1 worden de belangrijkste kenmerken visueel representeert.

2.3. Supermarktdata: webscraping als praktische pijplijn

Omdat Belgische retailers zelden API's voor product-/prijsfeeds openbaar maken, is webscraping een pragmatische manier om gestructureerde prijsgegevens van openbare pagina's te verkrijgen. Hoewel (Logos et al., 2023) en (Brown et al., 2024) een ethische en methodologische benadering van webscraping beschrijven, stellen ze geen specifieke technische implementatie voor voor gevallen waarin openbare API's niet beschikbaar zijn.
Voortbouwend op hun aanbevelingen wordt in dit onderzoek het volgende proces voorgesteld: HTML-opvraging, parsing van de content, headless browser voor JavaScript-afhankelijke content en opslag van de prijsgegevens. Deze aanpak voor de specifieke Belgische markt is geïnspireerd op het (Ken Van Loon, 2018) artikel van Statbel.

2.4. Juridische en ethische overwegingen voor scraping

Scraping moet voldoen aan de servicevoorwaarden (ToS), intellectuele eigendomsrechten en beperkingen op het gebied van gegevensbescherming. Vergelijkende analyses van de ToS van websites laten zien dat veel platforms "robots/scrapers" expliciet reguleren, waardoor onderzoekers noodzakelijkheid, proportionaliteit en nalevingsmechanismen moeten afwegen (Fiesler et al., 2020). Recente overzichten stellen concrete checklists voor over legaliteit, ethiek en institutionele beoordeling: bijvoorbeeld het documenteren van het doel, snelheidslimieten, opslag en datadeling (Brown et al., 2024; Logos et al., 2023). Deze kaders vormen de basis voor het beheer van het prototype.

2.5. Productmatching tussen retailers

Prijsvergelijking vereist het matchen van 'hetzelfde' artikel in alle winkels, ondanks verschillen in naamgeving/verpakkingsgrootte. Bestaande literatuur ondersteunt een tweefasenaanpak: 1. exacte identificatiegegevens (bijv. EAN/GTIN) indien beschikbaar; 2. benaderende/semantische matching met behulp van fuzzy similarity (Levenshtein/TF-IDF/cosinus) of ML-embeddings voor detectie van bijna-duplicaten (Kerek, 2020; Ning et al., 2022). Deze methoden koppelen de door de gebruiker op-

Tabel 2.1: Functionele vergelijking tussen bestaande tools en het voorgestelde prototype

Kenmerk	PingPrice	G4U	Prototype
Naam matching	-	?	+
Transparantie	-	-	+
Beperkingen (afstand/aantal winkels)	-	-	+
Gratis	+	-	+

+ = ondersteund, - = niet ondersteund, ? = gedeeltelijk/onduidelijk

gegeven productnaam direct aan het specifieke productaanbod van de winkel.

2.6. Beslissingsondersteuning, vertrouwen en beperkingen in boodschappenapps

Vertrouwen is een cruciale factor die de bereidheid van gebruikers om digitale boodschappentools te gebruiken beïnvloedt. (Chakraborty et al., 2024) benadrukt het belang van geloofwaardigheid van informatie, duidelijkheid en kwaliteit van de interactie om het vertrouwen van gebruikers in online boodschappenomgevingen te vergroten. Voortbouwend op dit perspectief benadrukt (DeZao, 2024) het vertrouwen in AI-gestuurde systemen. Door hun gegevensbronnen en tijdstempels te tonen, worden deze systemen transparanter en daardoor ook als betrouwbaarder en eerlijker ervaren door gebruikers. Bovendien beïnvloeden praktische beperkingen, zoals reisafstand en de mogelijkheid om slechts een bepaald aantal winkels te bezoeken, het nut van dergelijke tools. Integratie van deze beperkingen breidt de criteria voor beslissingsondersteuning verder uit en verbetert deze.
Samenvattend, de literatuur ondersteunt een pijplijn die webscraping, reproduceerbare matching (EAN-first + fuzzy/ML fallback) en transparante interfaces combineert, geëvalueerd op precisie/recall voor matches en realistische, op de student gerichte beperkingen (bijv. afstand, maximaal aantal winkels) voor kostenresultaten.

3

Methodologie

Dit proefschrift richt zich op het ontwerp en de implementatie van een functioneel prototype dat automatisch supermarktprijzen in Gent verzamelt, vergelijkt en matcht.

3.1. Proof of concept

De proof of concept opbouw bestaat uit vier opeenvolgende fasen: systeemontwerp, dataverzameling, dataverwerking en evaluatie.

3.1.1. Systeemontwerp

In de eerste fase werden de architectuur en datastroom van het systeem gedefinieerd met als doel modulariteit, schaalbaarheid en transparantie te garanderen.Het systeem is opgebouwd als een modulaire, event-gedreven pipeline bestaande uit vier lagen. De eerste laag is de scrapingslaag: product- en prijsinformatie wordt verzameld van de websites. De tweede laag is de transportlaag, verantwoordelijk voor publicatie van de ruwe data komende uit de scrapingslaag in Kafka. De derde laag is de verwerkingslaag, daar een Kafka-consumer verwerkt, normaliseert, matcht producten, en slaat de resultaten op in een PostgreSQL-database. De vierde laag is de presentatielaag: een Django-gebaseerde webinterface dat het mogelijk maakt om boodschappenlijsten en beperkingen in te voeren en berekent de goedkoopste winkelcombinatie.

3.1.2. Dataverzameling

De dataverzamelingsfase richt zich op het verzamelen van dagelijkse product- en prijsinformatie van geselecteerde Gentse supermarkten. Dit wordt uitgevoerd met behulp van webscrapingtechnieken, geïmplementeerd via Scrapy-spiders. Scrapy wordt gebruikt om HTTP-verzoeken te versturen en de onbewerkte HTML-inhoud

van webpagina's op te halen, waardoor toegang wordt verkregen tot publiek beschikbare informatie zonder een volledige browseromgeving.
De spiders extraheren relevante gegevens zoals productnamen, verpakkingsformaten, prijzen en merklabels door specifieke HTML-elementen te parsen. Voor websites die gebruikmaken van JavaScript-gedreven dynamische inhoud wordt een browser-emulator ingezet, zodat pagina's eerst volledig kunnen laden voordat ze verwerkt worden.
In plaats van de gegevens onmiddellijk op te slaan of te verwerken, worden alle gescrapete productrecords als ruwe JSON-objecten gepubliceerd naar een Kafka-topic. Hierdoor wordt de dataverzameling losgekoppeld van de daaropvolgende verwerkingsstappen, wat de schaalbaarheid en fouttolerantie van het systeem verhoogt. Naast de productinformatie zelf wordt ook metadata zoals tijdstip en bronwinkel meegestuurd, waardoor transparantie en reproduceerbaarheid worden gegarandeerd.

3.1.3. Dataverwerking en productmatching

De dataverwerking gebeurt asynchroon in een aparte module die berichten uit Kafka consumeert. Omdat supermarkten verschillende productnamen en -formaten gebruiken, moeten de verzamelde gegevens worden voorbewerkt voordat ze kunnen worden vergeleken. Deze fase bestaat uit een aantal stappen: data cleaning, normalisatie en productmatching. De dataopschoningsstap omvat het verwijderen van duplicaten en eenheidsnormalisatie (bijvoorbeeld prijs per kg of per liter).
De matchingstap implementeert string-gelijkaardigheidssalgoritmen, zoals Levehnstein-afstand en cosinus-similariteit op TF-IDF-vectoren, om gelijkwaardige producten in verschillende winkels te matchen. De filterstap slaat de dichtstbijzijnde productmatches op om nauwkeurigheid in vergelijkingen te garanderen. Het resultaat van deze fase is een uniforme dataset waarin identieke of vergelijkbare producten uit verschillende winkels direct kunnen worden vergeleken.

3.1.4. Interfaceontwikkeling en integratie

De tool cobineert alle componenten in één Django-gebaseerde webapplicatie. Daarnast wordt er een vergelijking module binnen deze applicatie uitgewerkt die berekent voor een ingevoerd boodschappenlijstje de kostprijs als alle producten in één winkel worden gekocht en de minimale totale prijs bij een optimale winkelcombinatie, rekening houdend met maximale reisafstand en maximaal aantal winkels.

3.1.5. Evaluatie

De evaluatiefase beoordeelt de praktische bruikbaarheid van het systeem, met behulp van vooraf gedefinieerde winkelwagentjes voor studenten die realistische aankoopscenario's simuleren. Elk winkelwagentje wordt vanuit twee perspectieven geanalyseerd: winkelen in één winkel (alle artikelen in één supermarkt kopen) en ge-

optimaliseerd winkelen in meerdere winkels (alle artikelen kopen op basis van de aanbevelingen van het systeem).
Op basis van deze resultaten kan het prototype worden beschouwd als een succesvol proof-of-concept als het in staat is om kostenbesparingen te realiseren voor Gentse studenten met verschillende criteria, terwijl de transparantie in het besluitvormingsproces behouden blijft.

4

Prototype

Het prototype dat in deze bachelorproef werd ontwikkeld, is een proof-of-cooncept van een web scraping applicatie die bruikbaar is voor een breed publiek - met name studenten - dat niet beschikt over uitgebreide kennis van programmeren of softwareontwikkeling. De gebruiker heeft slechts een internetverbinding nodig en hoeft geen technische configuraties uit te voeren.
Deze doelstelling impliceert dat de technische complexiteit van web scraping wordt volledig afgeschermd van de eindgebruiker. Interacties met het systeem gebeuren via een eenvoudige en intuïtieve gebruikersinterface, terwijl alle processen zoals het ophalen van webpagina's, het verwerken van data, het omgaan met anti-scrapingsmechanismen en de opslag van gegevens, achter de schermen verlopen. Het prototype fungeert dus als een laag tussen de gebruiker en de onderliggende scrapingsinfrastructuur, waarbij gebruiksvriendelijkheid en betrouwbaarheid centraal staan.

4.1. Keuze van programmeertaal en frameworks

De keuze van programmeertaal en bijhorende frameworks vormt een essentiële ontwerpbeslissing binnen dit prototype. Literatuur toont aan dat verschillende programmeertalen geschikt zijn voor web scraping, waaronder Python, JavaScript en C#. Op basis van vergelijkende studies en best practices (Journal, 2024; Khder, 2021; Majebi, 2025; ProWebScraper, 2018) is Python gekozen als primaire programmeertaal voor het prototype.
Deze keuze voor Python kan gemotiveerd worden door meerdere factoren. Ten eerste beschikt Python over een uitgebreid ecosysteem van bibliotheken die specifiek ontworpen zijn voor interactie met de inhoud van websites, zoals *Requests*, *BeautifulSoup*, *Scrapy* en *Playwright*. Hierdoor kan snel en efficiënt worden ingespeeld op uiteenlopende scrapingsuitdagingen, variërend van eenvoudige HTML-pagina's

tot complexe, dynamisch gegenereerde webinhoud. Ten tweede is Python relatief eenvoudig aan te leren, wat het bijzonder geschikt maakt voor beginnende ontwikkelaars en studenten. De leesbaarheid van de syntaxis en de grote hoeveelheid beschikbare documentatie en voorbeelden verlagen de instapdrempel aanzienlijk. Dit sluit aan bij de doelstelling van dit prototype, namelijk het ontwikkelen van een systeem dat toegankelijk is voor gebruikers zonder diepgaande programmeerkennis. Tot slot biedt Python voldoende mogelijkheden voor toekomstige uitbreidingen, zoals grootschalige dataverwerking, data-analyse en integratie van machine learning technieken. Hierdoor is Python niet alleen geschikt voor het huidige prototype, maar ook toekomstbestendig.

Binnen het Python-ecosysteem bestaan meerdere bibliotheken voor web scraping. Na een vergelijking werd gekozen voor *Scrapy* als centraal scrapingsframework. Deze keuze kan onderbouwd worden door het werk van (Eyzenakh et al., 2021), waarin verschillende scrapingsoplossingen zijn vergeleken op vlak van performantie, schaalbaarheid en architecturale opbouw. Scrapy onderscheidt zich van eenvoudigere oplossingen zoals *Requests* en *BeautifulSoup* door zijn asynchrone en event-gedreven architectuur. Dit maakt het mogelijk om meerdere webpagina's gelijktijdig te verwerken, wat resulteert in betere prestaties en efficiënter gebruik van systeembronnen. Daarnaast voorziet Scrapy ingebouwde ondersteuning voor request scheduling, foutafhandeling, throttling en uitbreidbare pipelines voor dataverwerking. Een bijkomend voordeel van Scrapy is de duidelijke scheiding tussen verschillende verantwoordelijkheden, zoals het ophalen van data, het parsen van responses en het verwerken van resultaten. Deze modulaire opbouw sluit nauw aan bij de architecturale principes die in de literatuur worden beschreven en verhoogt de onderhoudbaarheid van het systeem.

Hoewel Scrapy krachtig is voor het verwerken van statische HTML-pagina's, volstaat het niet in alle situaties. Steeds meer websites maken gebruik van JavaScript om inhoud dynamisch te genereren. Om deze pagina's correct te kunnen verwerken, werd *Playwright* geïntegreerd in het scrapingsproces. Playwright maakt het mogelijk om een echte browseromgeving te simuleren en JavaScript-code uit te voeren alvorens de HTML wordt geëxtraheerd. Hierdoor kunnen ook websites met complexe client-side logica succesvol gescrapet worden. De combinatie van Scrapy en Playwright zorgt voor een flexibele aanpak waarbij per website de meest geschikte scrapingstrategie kan worden toegepast.

Om de verschillende lagen binnen het systeem van elkaar te ontkoppelen, werd gekozen voor Apache Kafka als verbindende component tussen de scrapingslaag en verdere verwerking van data. Kafka fungeert hierbij als een message broker die ruwe scrapingsresultaten doorstuurt naar volgende verwerkingsstappen. De keuze voor Kafka is geïnspireerd door moderne, event-gedreven architecturen zoals beschreven in (Alexander, 2015). Door gebruik te maken van een message broker wordt de afhankelijkheid tussen scraping en verwerking verminderd. Dit ver-

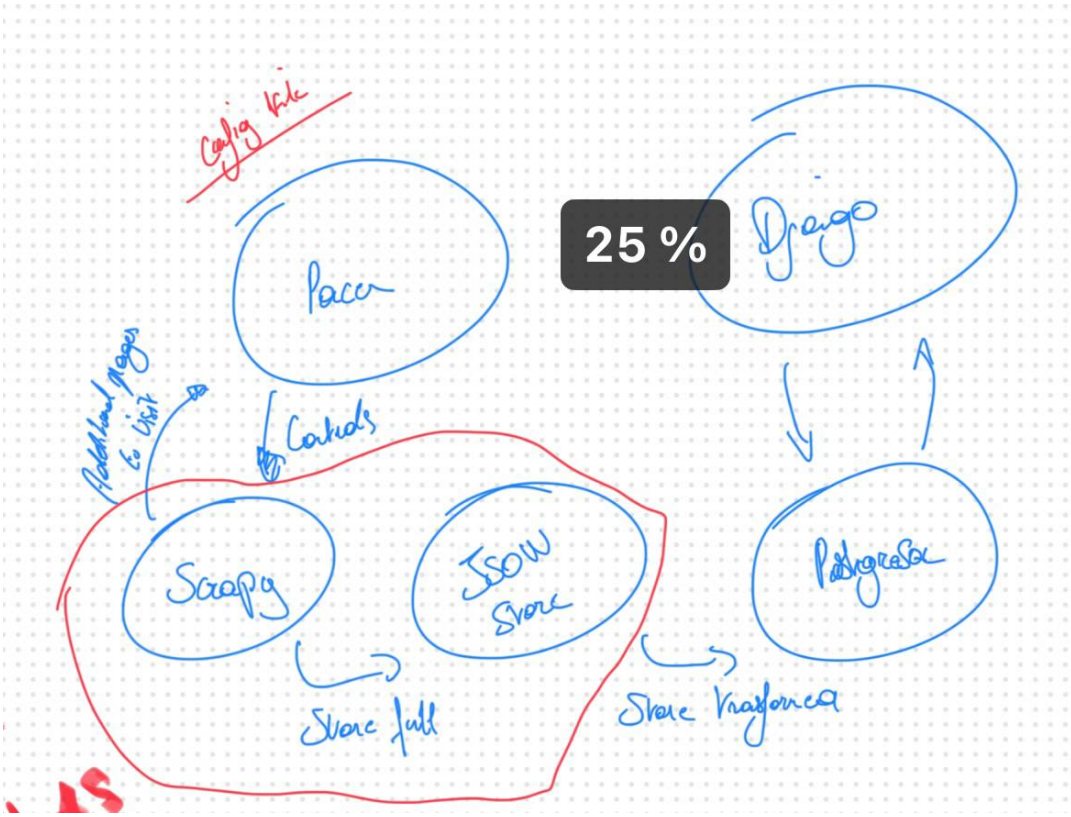
hoogt de schaalbaarheid en fouttolerantie van het systeem en maakt het mogelijk om data opnieuw te verwerken zonder de scrapingsfase te herhalen. Hoewel Kafka in dit prototype niet noodzakelijk op industriële schaal wordt ingezet, biedt het conceptueel een duidelijke meerwaarde en vormt het een solide basis voor toekomstige uitbreidingen.
Voor de backend van het prototype werd gekozen voor het Django-framework. Django sluit naadloos aan bij Python en biedt uitgebreide ondersteuning voor webapplicaties, databankintegratie en gebruikersinteractie. Een belangrijk voordeel van Django is de eenvoudige koppeling met Python-gebaseerde scrapingscomponenten. Hierdoor kunnen scrapingsprocessen, dataverwerking en gebruikersinterface binnen één technologisch ecosysteem worden geïntegreerd. Daarnaast voorziet Django standaardfunctionaliteiten zoals ORM (Object-Relational Mapping), authenticatie en administratie, wat de ontwikkeltijd aanzienlijk verkort.
Om de reproduceerbaarheid en consistentie van het systeem op verschillende machines te garanderen, werd gebruikgemaakt van Docker. Docker maakt het mogelijk om de volledige applicatie, inclusief afhankelijkheden, configuraties en services, te verpakken in containers. Hierdoor kan het prototype zonder bijkomende installatieproblemen uitgevoerd worden op verschillende systemen, wat zowel de ontwikkeling als de evaluatie vereenvoudigt.
De combinatie van Python, Scrapy, Kafka, Django en Docker resulteert in een coherente en uitbreidbare architectuur waarin elke technologie een duidelijk afgebakende rol vervult.

4.2. Klassieke web scraping pijplijn

De structuur van web scraping systemen wordt in de literatuur vaak beschreven als een opeenvolging van afzonderlijke verwerkingsstappen. (Laender et al., 2002) en (Khder, 2021) beschrijven web data extractie als een pijplijn bestaande uit meerdere logisch gescheiden fasen:

- selectie van databronnen;
- extractie van ruwe data;
- transformatie en normalisatie;
- integratie en opslag.

Deze architecturale scheiding verhoogt de onderhoudbaarheid van het systeem en maakt hergebruik en schaalbaarheid mogelijk. Het prototype volgt deze pijplijnstructuur expliciet. Op basis van de overleg met de co-promotor en de besproken state-of-the-art werd een modulaire architectuur voorgesteld. Deze architectuur is weergegeven in Figuur 4.1.
De architectuur bestaat uit vijf hoofdlagen:



Figuur 4.1: Overzicht van de voorgestelde prototype-architectuur

1. gebruikersinterface
2. scrapinglaag
3. opslag van ruwe data
4. transformatielaag
5. gestructureerde databank

Figuur 4.2.

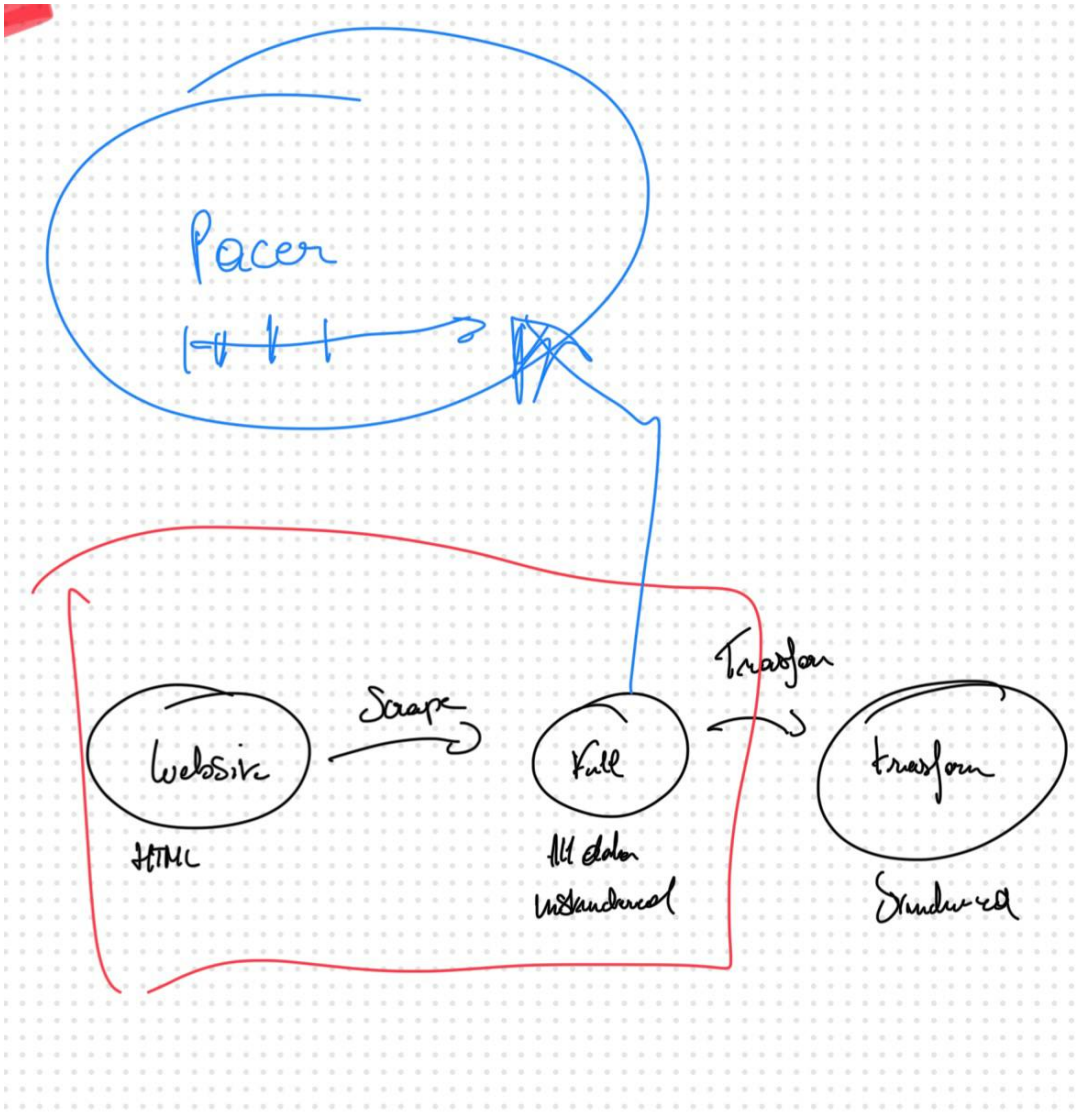
4.3. Gebruikersinterface

De gebruikersinterface vormt het enige contactpunt tussen de gebruiker en het systeem. De gebruiker kan via de interface aangeven welke website of productcategorie gescrapet moet worden. De verdere technische uitvoering wordt volledig door het systeem afgehandeld.

4.4. Scrapinglaag

4.4.1. Initiële aanpak en beperkingen

In een eerste experimentele fase werd gebruikgemaakt van de combinatie *Requests* en *BeautifulSoup*. Hoewel deze aanpak eenvoudig te implementeren is, bleek ze in

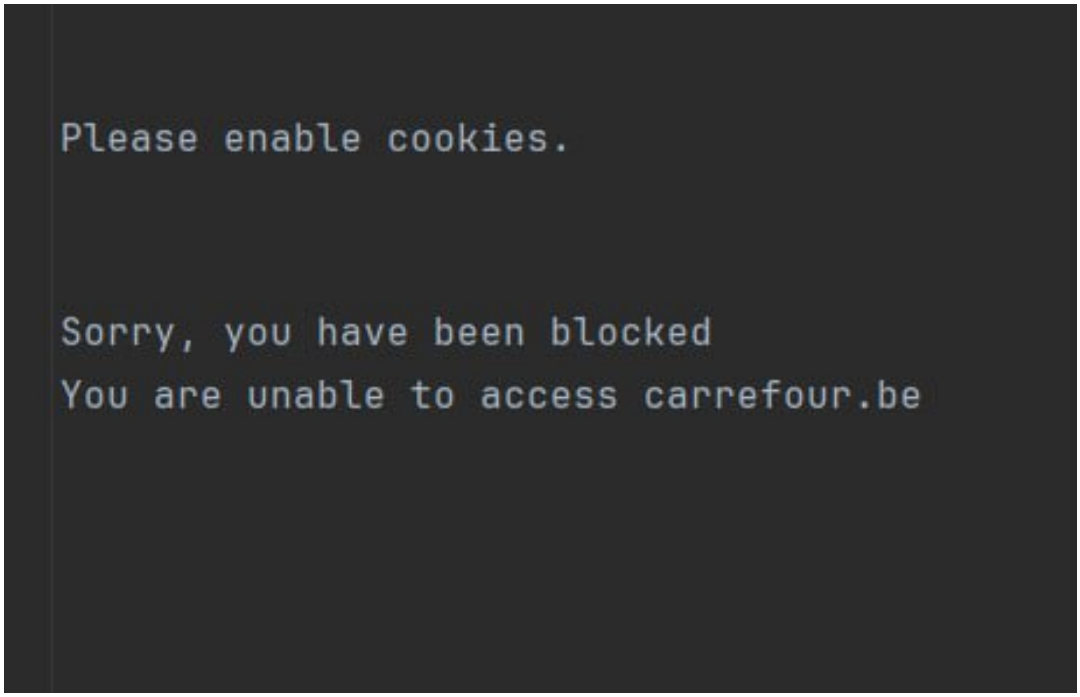


Figuur 4.2: Overzicht van de voorgestelde prototype-architectuur

de praktijk onvoldoende robuust. Tijdens tests werden meerdere HTTP-foutcodes waargenomen, waaronder: 403 (Forbidden), 456 (Quota exceeded). Bovendien werd vastgesteld dat bepaalde websites, zoals Carrefour, gebruikmaken van Cloudflare om niet-menselijk verkeer te detecteren en te blokkeren Figuur 4.3.

4.4.2. Overstap naar Scrapy en Playwright

Op advies van de co-promotor werd overgestapt naar het Scrapy-framework. Scrapy biedt uitgebreide mogelijkheden voor request scheduling, foutafhandeling en data-extractie, en is gebaseerd op een asynchrone, event-gedreven architectuur. Door het scrapingproces te mogen isoleren en als een afzonderlijke service of proces uit te voeren, wordt het systeem beter bestand tegen fouten en netwerkproblemen. Hierdoor blijft de rest van de applicatie operationeel, zelfs wanneer scraping taken falen of tijdelijk onderbroken worden. Daarnast om detectie te vermijden, wordt ge-



Figuur 4.3: Gedetecteerd met BeautifulSoup

bruikgemaakt van *scrapy-impersonate*, waarmee browserheaders en gebruikersgedrag geïmiteerd kunnen worden. Voor websites met dynamische content werd Scrapy gecombineerd met Playwright, zodat JavaScript-elementen correct kunnen worden geladen.

4.4.3. Website-specifieke scrapingstrategieën

Voor de verschillende webshops werden specifieke scrapingstrategieën toegepast:

- **Albert Heijn**: data wordt opgehaald via de GraphQL-API, waarbij JSON-responses rechtstreeks worden verwerkt (JaapWestera, z.d.).
- **Carrefour**: HTML-scraping met Scrapy en browser-imitatie.
- **Colruyt**: scraping met Scrapy in combinatie met Playwright om JavaScript-gegenereerde content te laden.

Deze aanpak minimaliseert blokkades en verhoogt de betrouwbaarheid van het scrapingproces.

4.5. Opslag van ruwe data

Alle opgehaalde data wordt initieel opgeslagen als ruwe, ongestructureerde data. Deze keuze biedt meerdere voordelen. Ten eerste kan data opnieuw verwerkt worden zonder opnieuw te scrapen. Ten tweede blijft de oorspronkelijke brondata beschikbaar voor validatie en foutanalyse.

Deze stap sluit conceptueel aan bij event-gedreven architecturen waarbij data eerst als onbewerkte events wordt opgeslagen.

4.6. Transformatielaag

De transformatielaag is verantwoordelijk voor het omzetten van ruwe data naar een gestructureerd formaat. In deze fase worden onder andere:

- productnamen opgeschoond;
- prijsnotaties genormaliseerd;
- eenheden geharmoniseerd;
- irrelevante HTML-elementen verwijderd.

Door transformatie los te koppelen van scraping wordt het systeem beter onderhoudbaar en uitbreidbaar.

4.7. Gestructureerde opslag

Na transformatie wordt de data opgeslagen in een relationele databank, in dit prototype PostgreSQL. Deze databank vormt de basis voor verdere analyse, prijsvergelijkingen en mogelijke visualisaties.

4.8. Conclusie

Het prototype combineert een gebruiksvriendelijke interface met een modulaire en schaalbare backend-architectuur. Door de scheiding tussen scraping, opslag en transformatie sluit het ontwerp nauw aan bij zowel academische literatuur als hedendaagse best practices.
Deze architectuur biedt een solide basis voor verdere uitbreidingen, zoals automatische planning van scrapingtaken, ondersteuning voor bijkomende websites en grootschalige data-analyse.

5

Testen en Resultaten

Om de effectiviteit van het prototype en het onderliggende algoritme te evalueren, werd gekozen voor een praktijkgerichte testmethode gebaseerd op het vergelijken van winkelmanden. Deze aanpak sluit nauw aan bij het beoogde gebruiksscenario van de applicatie, namelijk het vergelijken van productprijzen tussen verschillende winkels.
Een winkelmand wordt hierbij gedefinieerd als een verzameling producten die door een gebruiker geselecteerd worden. Voor elke winkelmand werd het scraping- en verwerkingsproces volledig doorlopen, waarna de totale prijs per webshop werd berekend op basis van de beschikbare productdata.

6

Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetuer libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem. Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

A

Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1. Introduction

In recent years, food prices in Belgium have risen significantly, placing increasing financial pressure on students and other budget-conscious consumers. While several supermarket price comparison tools exist, they generally focus on presenting the lowest prices without considering practical limitations, such as the distance a consumer is willing to travel or the number of stores they can reasonably visit. As a result, these tools provide theoretically optimal solutions that are difficult to implement in real-world scenarios, particularly for students with limited mobility and tight schedules.
Students frequently face challenges in identifying the most cost-effective shopping options. Limited budgets, combined with time and travel constraints, complicate efficient price comparisons across different supermarkets. Existing tools rarely incorporate these constraints, creating a gap between available price information and actionable, user-centered insights. This research addresses this gap by focusing on the development of a system tailored to the needs of students in Ghent.

To develop such a system, the following main research question needs to be answered: How to design and develop a transparent and scalable system, capable of automatically collecting, matching, and comparing supermarket prices in Ghent, that takes into account consumers' distance and limits the amount of shops visited?
To support answering this question and guide the research, an improved understanding of the target audience and the problem context is required. More specifi-

cally:

- What factors currently influence students' consumption habits in Belgium?
- What kind of tools for price comparisons are available in Belgium, and what shortcomings do they have for students?
- What technical and practical challenges are there for collecting price data from Belgian supermarkets?

Furthermore, research into programmatic and architectural details of the candidate solution, and its success factors, needs to be conducted. More precisely:

- How can supermarket price data be automatically collected and structured?
- What approaches can be used to match general product names to achieve accurate comparisons?
- How can the system calculate and recommend the most cost-effective combinations of stores within a user-defined distance and a store limit?
- How can the system architecture be designed to support scalability and transparency of the data?
- What criteria must the prototype meet in order to be considered a valid proof-of-concept?

The result of this research is a prototype, implemented using Python and Django, that collects price data through web scraping from Belgian supermarket websites. Users of this prototype will be able to input a general shopping list and specify a maximum travel distance or a limit on the number of stores. The system will then calculate the most cost-effective combination of stores based on these constraints. The system will be evaluated using a predefined "student shopping cart" to compare the total cost of shopping at a single store versus the optimized multi-store recommendation generated by the system.

This research contributes to the development of a realistic and accessible supermarket price comparison tools that integrate technical efficiency with consumer-centered constraints. By focusing on students' needs, the system aims to enhance price transparency and support informed, budget-conscious shopping decisions, offering practical insights that could inform future consumer-oriented applications.

A.2. Literature Review

A.2.1. Context: food prices and student pressure

Recent Belgian indicators show persistent price pressure on food. According to Statbel's CPI' report, the headline and core inflation remain elevated throughout

2024-2025, with core inflation above 2% in October 2025(Statbel, 2025a, 2025b). Independent tracking by Testaankoop/Testachats likewise reports supermarket specific inflation
around 4% in 2025 (Testaankoop, 2025a, 2025b). Broader macro assessments (*OECD Economic Surveys: Belgium 2024*, 2024) show the detailed impact of inflation, confirming price pressure on consumers. Together, these sources substantiate the problem relevance for the price-sensitive groups such as students.

A.2.2. Existing solutions

There are several Belgian specific tools available to help consumers compare supermarket prices and select better products such as PingPrice (PingPrice, 2024) and G4U (G4U, 2025). However, both apps have their limitations. PingPrice compares products using barcodes, which prevents it from effectively comparing store-brand or generic products that lack standardized identifiers. As a result, many relevant items are excluded from comparisons.
G4U, on the other hand, offers extensive product and promotion information, but operates as a paid service, limiting its accessibility for students who already face financial constraints. Consequently, there is a need for a free and transparent alternative that allows users to compare generic product categories, rather than barcodes.

A.2.3. Supermarket data: web scraping as a practical pipeline

Because Belgian retailers rarely expose APIs for product/price feeds to the public, web scraping is a pragmatic way of obtaining structured price data from public pages. While (Logos et al., 2023) and (Brown et al., 2024) guide through ethical and methodological approach to web scraping, they do not propose specific technical implementation for cases where public APIs are not available.
Building on their recommendations, the following process is proposed in this paper: HTML retrieval, parsing of the content, automated browser rendering for JavaScript-dependent content, and storing the price data. This approach to Belgian specific market is inspired by Statbel (Ken Van Loon, 2018) paper.

A.2.4. Legal and Ethical considerations for scraping

Scraping must respect terms of service (ToS), IP, and data-protection constraints. Comparative analyses of website ToS show many platforms explicitly regulate ”robots/scrapers”, requiring researchers to weigh necessity, proportionality, and compliance mechanisms (Fiesler et al., 2020). Recent overviews propose concrete checklists on legality, ethics, and institutional review: for example, documenting purpose, rate limits, storage, data sharing (Brown et al., 2024; Logos et al., 2023). These frameworks guide the governance of the prototype.

A.2.5. Product matching across retailers

Price comparison requires aligning "the same" item across stores despite naming/pack size differences. Literature supports a two-stage approach: 1. exact identifiers (e.g., EAN/GTIN) where available; 2. approximate/semantic matching using fuzzy similarity (Levehnstein/TF-IDF/cosine) or ML embeddings for near-duplicate detection (Kerek, 2020; Ning et al., 2022). These methods map directly from user-supplied product name to store's specific unit.

A.2.6. Decision support, trust, and constraints in grocery apps

Trust is a critical factor influencing user's willingness to adopt digital grocery tools. (Chakraborty et al., 2024) highlights the importance of information credibility, clarity and quality of interaction for building user's trust in online grocery environments. Building on this perspective, (DeZao, 2024) emphasizes trust in AI-powered systems. By showing their data sources and timestamps, these systems become more transparent and consequently are perceived as more reliable and fair by users. Moreover, real-world constraints such as travel distance and ability to visit certain amount of stores affect the usefulness of such tools. Integration of these constraints expands and improves decision support criteria.
In summary, the literature supports a pipeline combining web scraping, reproducible matching (EAN-first + fuzzy/ML fallback), and transparent interfaces that expose source and recency, evaluated on precision/recall for matches and realistic student-centric constraints (e.g. distance, maximum amount of stores) for cost outcomes.

A.3. Methodology

This thesis focuses on the design and implementation of a functional prototype that automatically collects, matches, and compares supermarket prices in Ghent. The research process is divided into five main phases: system design, data collection, data processing and product matching, system implementation and evaluation.

A.3.1. System Design

In the first phase, the system's architecture and data flow were defined to ensure modularity, scalability, and transparency. The architecture consists of three distinct layers. The first layer is called the data layer and manages the storage of product and price information in a PostgreSQL relational database. The second layer is the processing layer and handles web scraping, data cleaning, and product matching logic, implemented in Python. The third layer is the presentation layer, which provides a user interface through a Django web application, allowing users to input shopping lists and define constraints such as travel distance and the number of stores. These design choices were made to support future scalability and the integration of new supermarket and product data.

A.3.2. Data Collection

The data collection phase focuses on gathering daily product and price information from selected Ghent supermarkets. This is achieved using web scraping techniques, using Python libraries such as Requests, BeautifulSoup, and Selenium. Requests is used to send HTTP requests and retrieve the raw HTML content of web pages, giving access to publicly available information without a browser. BeautifulSoup is employed to parse and extract specific elements from the HTML content obtained through Requests. Lastly, Selenium is used for scraping dynamic websites that load data through JavaScript after initial page request.
Each scraper retrieves product names and prices. All data is stored along with additional metadata such as timestamps and data sources to maintain transparency and traceability.

A.3.3. Data Processing and Product Matching

Since supermarkets use different product names and formats, the collected data requires preprocessing before comparison. This phase happens in a number of steps: data cleaning, normalization and product matching. The data cleaning step includes the removal of duplicates and unit normalization (e.g. price per kg or per liter). The matching step implements string similarity algorithms, such as Levehnstein distance and cosine similarity on TF-IDF vectors, to match equivalent products across stores. The filtering step stores the closest product matches to ensure accuracy in comparisons. The result of this phase is a unified dataset where identical or similar products from different stores can be compared directly.

A.3.4. System implementation

The tool integrates following components into a single Django-based web application. Data Scraper runs scheduled scraping jobs and updates the database. Data Processor performs data cleaning and product matching. Comparison Engine calculates the most cost-effective store combinations based on user's constraints. User Interface allows users to input shopping lists and define parameters such as maximum distance and store count. The system is designed for local deployment during testing but can be extended for public use.

A.3.5. Evaluation

The evaluation phase assesses the practical usefulness of the system, using predefined student shopping carts that simulate realistic purchase scenarios. Each cart is analyzed from two perspectives: Single-store shopping (purchasing all items in one supermarket) and Optimized multi-store shopping (purchasing all items using the system's recommendation).
Based on these results, the prototype can be considered a successful proof-of-concept if it is capable of producing cost reductions for Ghent students with various criteria,

while maintaining transparency in its decision process.

A.4. Expected results

The main expected outcome of this research is a functional prototype that demonstrates the feasibility of a system for collecting, matching and comparing product prices. This includes a working scraping module that collects price and data from multiple stores' websites, and a matching module for corresponding items across different shops with a high level of accuracy. After evaluation using predefined student carts, the prototype is expected to demonstrate measurable price difference when compared to shopping in a single supermarket. These results should confirm added value of the system in terms of affordability.

Bibliografie

Alexander, S. (2015). Distributed Frontera: Web crawling at scale. *Zyte*. https://www.zyte.com/blog/distributed-frontera-web-crawling-at-large-scale

Brown, M. A., Gruen, A., Maldoff, G., Messing, S., Sanderson, Z., & Zimmer, M. (2024). Web Scraping for Research: Legal, Ethical, Institutional, and Scientific Considerations. https://doi.org/10.48550/ARXIV.2410.23432

Chakraborty, D., Kumar Kar, A., Patre, S., & Gupta, S. (2024). Enhancing trust in online grocery shopping through generative AI chatbots. *Journal of Business Research*, *180*, 114737. https://doi.org/10.1016/j.jbusres.2024.114737

DeZao, T. (2024). Enhancing transparency in AI-powered customer engagement. *Journal of AI, Robotics & Workplace Automation Volume 3 Number 2,2024*. https://doi.org/10.48550/ARXIV.2410.01809

Eyzenakh, D., Rameykov, A., & Nikiforov, I. (2021). High Performance Distributed Web-Scraper. *Proceedings of the Institute for System Programming of the RAS*, *33*(3), 87–100. https://doi.org/10.15514/ispras-2021-33(3)-7

Fiesler, C., Beard, N., & Keegan, B. C. (2020). No Robots, Spiders, or Scrapers: Legal and Ethical Regulation of Data Collection Methods in Social Media Terms of Service. *Proceedings of the International AAAI Conference on Web and Social Media*, *14*, 187–196. https://doi.org/10.1609/icwsm.v14i1.7290

G4U. (2025). G4U. https://g4u-app.com/

JaapWestera. (z.d.). Appy. https://github.com/JaapWestera/albert-heijn-graphql-api/tree/main?tab=readme-ov-file

Journal, D. (2024). Javascript vs. Python for Web Scraping. *Data Journal*. https://medium.com/@datajournal/javascript-vs-python-for-web-scraping-dee9102e56cf

Ken Van Loon, D. R. (2018). Webscraping, de verzameling en verwerking van online data voor de consumptieprijsindex. https://statbel.fgov.be/sites/default/files/files/documents/Analyse/NL/webscraping_nl.pdf

Kerek, H. (2020). *Product Similarity Matching for Food Retail using Machine Learning* (**publication**Nr. 2020:067) [masterscriptie, KTH, Mathematical Statistics].

Khder, M. A. (2021). Web Scraping or Web Crawling: State of Art,Techniques, Approaches and Application. https://www.i-csrs.org/Volumes/ijasca/2021.3.11.pdf

Laender, A. H. F., Ribeiro-Neto, B. A., da Silva, A. S., & Teixeira, J. S. (2002). A brief survey of web data extraction tools. *ACM SIGMOD Record*, *31*(2), 84–93. https://doi.org/10.1145/565117.565137

Logos, K., Brewer, R., Langos, C., & Westlake, B. (2023). Establishing a framework for the ethical and legal use of web scrapers by cybercrime and cybersecurity researchers: learnings from a systematic review of Australian research. *International Journal of Law and Information Technology*, *31*(3), 186–212. https://doi.org/10.1093/ijlit/eaad023

Majebi, I. (2025). Best Practices for Web Scraping in 2025. https://www.scraperapi.com/web-scraping/best-practices/

Ning, W., Cheng, R., Shen, J., Haldar, N. A. H., Kao, B., Yan, X., Huo, N., Lam, W. K., Li, T., & Tang, B. (2022). Automatic Meta-Path Discovery for Effective Graph-Based Recommendation. *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 1563–1572. https://doi.org/10.1145/3511808.3557244

OECD Economic Surveys: Belgium 2024. (2024, september). OECD Publishing. https://doi.org/10.1787/c671124e-en

PingPrice. (2024). PingPrice. https://www.pingprice.be/en/

ProWebScraper. (2018). The 5 Best Programming Languages for Web Scraping. https://prowebscraper.com/blog/best-programming-language-for-web-scraping/

Statbel. (2025a). Harmonised index of consumer prices - December 2024. https://statbel.fgov.be/en/news/harmonised-index-consumer-prices-december-2024

Statbel. (2025b). Consumer price index - September 2025. https://statbel.fgov.be/en/themes/consumer-prices/consumer-price-index

Testaankoop. (2025a). Rundsvlees wordt duur: biefstuk 18% duurder dan vorig jaar, americain natuur 17% duurder. https://www.test-aankoop.be/familie-prive/supermarkten/pers/inflatie-april-2025

Testaankoop. (2025b). Inflatie in de supermarkt: net onder 4 % in september. *Helena Coupette*. https://www.test-aankoop.be/familie-prive/supermarkten/nieuws/maandelijkse-inflatie-supermarkt