**Jira / Final Comment – Category Normalization & Friendly Labels**

**Status:** Done ✅

**Owner:** Metehan / Core Team

**Epic link:** Frontend & Admin Evolution (Fase 3 – UX consistentie)

**Date:** 26-10-2025

**1. Probleem**

Tot nu toe kwamen categorieën in de app direct uit de database / AI-pipeline en waren dus inconsistent en niet UX-proof.

Concreet voorbeelden uit productiegegevens:

* "bakkal/supermarket"
* "barbershop"
* "fast\_food"
* "travel agency"

Issues die dit veroorzaakte:

* Filtering brak. Voorbeeld: locaties met categorie "bakkal/supermarket" verschenen niet onder de filter “Supermarket”, terwijl dit in de praktijk wel gewoon Turkse supermarkten/bakkals zijn.
* "barbershop" viel niet onder de filter “Barber”.
* UI liet lelijke labels zien zoals Travel\_agency en fast\_food (underscore en Engels), in plaats van nette user-facing labels zoals “Reisbureau” en “Fastfood”.
* Frontend moest gokken wat een categorie “is”, wat foutgevoelig is zodra de AI weer een nieuwe variant bedenkt.

Dit beïnvloedde direct de geloofwaardigheid van de app: gebruikers selecteren bijvoorbeeld “Supermarkt” en zien dan geen “YASAM SUPERMARKT”, “Sahan Supermarket”, “Ali Baba”, etc. Dat voelt alsof onze data leeg is.

We moesten dus naar één bron van waarheid voor categorieën, zowel voor weergave als filtering.

**2. Wat is gebouwd**

✅ **Nieuwe canonical categorie-laag in de backend**

We hebben een nieuwe module toegevoegd:

Backend/app/services/category\_map.py

Belangrijk hierin:

* CATEGORY\_MAP: een mapping van canonical categorieën naar:
  + een nette label voor de gebruiker (bijv. "Supermarkt", "Barber", "Reisbureau", "Moskee", "Fastfood", "Bakkerij"),
  + een lijst met alle synoniemen/ruwe varianten die we in het wild tegenkomen (bijv. "bakkal/supermarket", "bakkal", "turkse supermarkt", "barbershop", "haarmode", "fast food", "snackbar", "camii", "moskee").
* \_slugify() en \_titlecase\_label() helpers om veilige keys en nette labels te genereren als fallback.
* normalize\_category(raw: str):
  + Neemt de ruwe DB-category.
  + Zoekt of die matcht met één van onze bekende varianten.
  + Retourneert een dict met drie velden:
    - category\_raw: exact wat er in de database staat (bv. "bakkal/supermarket").
    - category\_key: onze canonical key (bv. "supermarket").
    - category\_label: nette label voor UI (bv. "Supermarkt").

We precomputen ook een lookup zodat "bakkal/supermarket" automatisch onder "supermarket" valt en "barbershop" onder "barber", enz.

Dit is future-proof: als de AI morgen “Turkse supermarkt” of “hair salon” zet, hoeven we geen code aan te raken — we voegen dat gewoon toe als extra match in de map.

✅ **API-output verrijkt**

In Backend/api/routers/locations.py (GET /api/v1/locations) hebben we de response van elke locatie verrijkt met die canonical info.

Elke locatie bevat nu:

* category\_raw → bv. "bakkal/supermarket"
* category\_key → bv. "supermarket"
* category\_label → bv. "Supermarkt"

We casten ook lat/lng/rating netjes naar numbers, houden id als string voor React, en we hebben defensieve fallback logic (zodat de API niet crasht als normalisatie ooit faalt).

Belangrijk:

We hebben NIETS in de database hoeven migreren. Dit zit 100% in de response-shape. Dus backwards compatible en veilig.

✅ **Frontend types bijgewerkt**

In Frontend/src/types/location.ts is het Location type uitgebreid met:

category\_raw?: string;

category\_key?: string;

category\_label?: string;

category (oude veld) blijft bestaan voor backward compatibility, maar wordt niet meer getoond in de UI.

✅ **Frontend rendering en filtering aangepast**

Voor weergave:

* Componenten zoals LocationCard tonen nu location.category\_label als primaire categorie-tekst.
* Geen underscores of rare Engelse strings meer voor de gebruiker.

Voor filtering:

* De frontend gebruikt nu category\_key als de interne filterwaarde.

Dus als de user “Supermarkt” kiest, de filter stuurt supermarket (canonical) naar de backend in de query.

* Alle varianten (“bakkal/supermarket”, “bakkal”, “turkse supermarkt”, …) mappen nu naar category\_key = supermarket, dus ze verschijnen allemaal in dezelfde lijst en op dezelfde kaartweergave.

Resultaat:

* Alle bakkals/supermarkten vallen nu onder één filter “Supermarkt”.
* Alle barbershops vallen nu onder “Barber”.
* “Travel\_agency” toont als “Reisbureau”, “fast\_food” toont als “Fastfood”.
* De gebruiker ziet nooit meer Travel\_agency of fast\_food met underscores in de UI.

**3. Acceptatiecriteria (alle gehaald)**

* Een locatie met category = "bakkal/supermarket" verschijnt wanneer ik filter op “Supermarkt”. ✅
* Een locatie met category = "barbershop" verschijnt wanneer ik filter op “Barber”. ✅
* UI toont nette labels zonder underscores of slash (“Reisbureau”, “Fastfood”, “Moskee”). ✅
* De backend response levert zowel ruwe als genormaliseerde categorievelden, zodat we altijd audit-trace houden (we verliezen geen informatie). ✅
* Frontend gebruikt category\_key voor state + filtering, category\_label voor tonen. ✅

**4. Definition of Done**

* Backend/app/services/category\_map.py bestaat en bevat CATEGORY\_MAP, \_slugify, \_titlecase\_label, en normalize\_category.
* /api/v1/locations voegt category\_raw, category\_key, category\_label toe aan elk record in de response.
* Frontend/src/types/location.ts kent deze nieuwe velden.
* Frontend componenten (o.a. de kaartlijstkaart / LocationCard) tonen category\_label en niet meer de ruwe DB-string.
* Filtering in de frontend werkt met category\_key (canonical), niet met willekeurige AI/pipeline strings.
* UX is nu consistent en begrijpelijk voor eindgebruikers.

✅ Story kan gesloten.

**Wat nu belangrijk is voor volgende stories**

1. **Nieuwe categorieën toevoegen = config, niet code**
   * Als we straks bv. “Notaris”, “Advocaat”, “Hammam”, “Bruidssalon”, “Café” toevoegen:
     + Alleen CATEGORY\_MAP uitbreiden met een nieuwe entry of extra synoniemen in match.
     + Frontend krijgt het automatisch goed (label netjes, filter werkt).
2. **AI mag creatief blijven**
   * We hoeven de AI niet hard te beperken.
   * Ook al noemt de AI iets “Turkse buurtbakkal”, zolang we dat woord toevoegen in de match-lijst onder "supermarket", wordt het netjes “Supermarkt” voor de user.
3. **Frontend mag altijd vertrouwen op category\_label**
   * Designers hoeven niet meer te hacken met replace('\_', ' ').
   * UX-teksten hoeven niet meer aangepast als de AI rare lowercase namen spuwt.
4. **Roadmap-impact**
   * Deze canonical laag is fundering voor:
     + Fase 3 / Frontend & Admin Evolution (The New Testament II, Fase 3)
     + Search & Filters Redesign (F3-S4 in The New Testament II Backlog) waar we categorie-iconen gaan tonen en een zoekbalk met suggesties bouwen.
   * Nu we een vaste category\_key hebben, kunnen we probleemloos iconen en kleuren koppelen per categorie en dat consistent tonen in zoekchips, bottom sheet, map markers, enz.

**Hoe je Cursor hierna inzet**

Je hebt Cursor nu al deze change laten doen. Top.

Volgende stap die je aan Cursor kunt geven (optioneel, niet urgent maar nice follow-up):

* “Gebruik category\_key om per categorie een standaard icoon en kleur te kiezen in de map markers en filterchips. Voeg (of update) een centrale mapping category\_icon\_map (we hebben hier al een tabel voor in de DB volgens de master plan en schema’s) zodat ‘supermarket’, ‘barber’, ‘bakery’, etc. elk een eigen icoon en kleur krijgen, en render die in de UI.”

Dat is basically F3-S4 groundwork.

Als je wilt kan ik daar de volgende Cursor-opdracht ook voor je uitschrijven.