UITBREIDINGEN OP HET SMOELENBOEK

Individuele PHP-programmeeropdrachten met technische uitwerking, documentatie, presentatie en mondelinge verdediging

In dit document vind je een reeks individuele uitbreidingen op de bestaande smoelenboekapplicatie. Elke uitbreiding vertrekt vanuit een concrete functionele opdracht en wordt uitgewerkt aan de hand van duidelijke doelstellingen, minimale vereisten, technische verwachtingen, uitbreidingsmogelijkheden en mondelinge kernvragen.

De leerling bouwt de uitbreiding zo zelfstandig mogelijk in, documenteert het proces zorgvuldig, gebruikt AI enkel als doordacht hulpmiddel en licht het eindresultaat toe in een presentatie met live demonstratie voor de klas.

Stijn Willekens

Inhoudsopgave

Uitbreidingen smoelenboek

Algemene richtlijnen voor alle uitbreidingen

Werk deze uitbreiding zo zorgvuldig en zo correct mogelijk uit binnen het bestaande smoelenboek. Het is de bedoeling dat je niet zomaar iets toevoegt, maar dat je nadenkt over hoe jouw uitbreiding logisch past binnen de huidige applicatie, databank en code-structuur.
Probeer dus **verzorgd** te **werken**, **duidelijke keuzes** te maken en je **oplossing degelijk** te **testen**.

AI als hulpmiddel

Je mag gebruik maken van **AI** als **hulpmiddel**, maar AI mag nooit zomaar jouw werk overnemen zonder dat je begrijpt wat er gebeurt. Probeer de uitbreiding eerst zelf zo ver mogelijk uit te werken. Als je vastloopt, mag je AI gebruiken om technieken, mogelijke aanpakken, voorbeeldcode of oplossingen te zoeken. Wat je uiteindelijk in je project opneemt, moet je wel volledig begrijpen. Je moet dus kunnen uitleggen wat de code doet, waarom je die oplossing gekozen hebt en hoe ze werkt binnen jouw applicatie.

Als je AI gebruikt tijdens het uitwerken van je uitbreiding, moet je daarvan ook een **logboek** bijhouden. In dat logboek noteer je welke prompts je gebruikt hebt, welke antwoorden of suggesties je van de AI gekregen hebt en wat je daar uiteindelijk wel of niet van hebt toegepast in je project. Voeg ook kort toe waarom een antwoord bruikbaar was, waarom je het aangepast hebt of waarom je het niet gebruikt hebt. Het doel van dit logboek is om transparant te maken hoe je AI als hulpmiddel hebt ingezet tijdens je denk- en ontwikkelproces.

Zinvolle commentaar

Voorzie in je code voldoende en **zinvolle commentaar**. Commentaar moet niet elke lijn beschrijven, maar wel helpen om belangrijke stukken code, moeilijke logica, databankkoppelingen, filters, query's of externe koppelingen duidelijk te maken.
Iemand die jouw code leest, moet kunnen volgen wat je gebouwd hebt.

Presentatie van jouw uitbreidging

Van deze uitbreiding maak je ook een (korte) **presentatie** voor de klas.
Daarin leg je uit:

- Wat je precies hebt gebouwd
- Hoe jouw uitbreiding werkt
- Welke technische keuzes je gemaakt hebt
- Hoe je dit hebt geïmplementeerd
- Welke problemen of moeilijkheden je bent tegengekomen
- Hoe je die problemen hebt opgelost, of waarom iets nog moeilijk blijft binnen deze applicatie

Daarnaast geef je ook een **kleine live demonstratie** van je uitbreiding. Je toont daarbij in de applicatie zelf wat werkt, hoe de uitbreiding gebruikt wordt en welk effect jouw aanpassing heeft op de werking van het smoelenboek.

De bedoeling van deze presentatie is niet alleen dat je toont wat werkt, maar vooral dat je bewijst dat je begrijpt wat je gebouwd hebt en dat je dit ook helder kan uitleggen aan anderen.

Uitbreiding woonplaatsen, geocoding en Leaflet-kaart

Korte opdrachtformulering

Maak een uitbreiding op het bestaande smoelenboek waarbij woonplaatsen niet langer als vrije tekst per collega worden opgeslagen, maar in een aparte tabel terechtkomen. Koppel collega's via een `city_id` aan die tabel, vraag voor nieuwe woonplaatsen automatisch coördinaten op via geocoding, en toon op een aparte pagina een `Leaflet`-kaart met per gemeente een marker en het aantal collega's in die gemeente.

Doel

Breid het bestaande smoelenboek uit met een kaartweergave van collega's per gemeente. Normaliseer de woonplaatsen in de databank, zodat gemeenten niet langer als vrije tekst bij elke collega staan, maar centraal beheerd worden. Toon op de publieke pagina een Leaflet-kaart met per gemeente één marker en het aantal collega's in die gemeente.

Databankaanpassingen

Voer minstens deze wijzigingen uit:

- Maak een nieuwe tabel `smoelboek_places` aan.
- Voorzie daarin minstens: `id`, `name`, `postal_code`, `country_code`, `latitude`, `longitude`, `created_at`, `updated_at`.
- Vervang in `smoelboek_colleagues` het veld `city` door `city_id`.
- Leg een foreign key-relatie van `smoelboek_colleagues.city_id` naar `smoelboek_places.id`.
- Zorg dat bestaande data gemigreerd kan worden of beschrijf hoe nieuwe data vanaf nul correct wordt opgeslagen.

Functionele eisen

De uitbreiding moet minstens dit gedrag hebben:

- Bij het toevoegen of bewerken van een collega kiest of vult de admin een woonplaats in.
- Het systeem controleert of die woonplaats al bestaat in `smoelboek_places`.
- Bestaat de woonplaats al, dan wordt de bestaande `city_id` gekoppeld aan de collega.
- Bestaat de woonplaats nog niet, dan vraagt het systeem via Nominatim de coördinaten op.
    - `https://nominatim.openstreetmap.org/search?postalcode=2440&country=Belgium&format=jsonv2&limit=1`
- De coördinaten van de nieuwe woonplaats worden daarna opgeslagen in `smoelboek_places`.
- Op een publieke map.php pagina komt een Leaflet-kaart.
- Op de kaart staat per gemeente één marker.
- De marker toont het aantal collega's in die gemeente.
- Bij klik op een marker verschijnt een popup met minstens de gemeentenaam en het aantal collega's.

Minimumvereisten

Een oplossing is pas voldoende als minstens aan dit alles voldaan is:

- De databank is genormaliseerd: woonplaatsen zitten in een aparte tabel.
- Collega's worden correct gekoppeld via `city_id`.
- Er is een werkende geocode-oproep naar Nominatim voor nieuwe woonplaatsen.
- De geocode wordt slechts uitgevoerd als de woonplaats nog niet bestaat.
- De kaart werkt met Leaflet en toont gegroepeerde markers per gemeente.
- De markerpopup toont correcte gegevens uit de databank.

- Er is foutafhandeling als Nominatim geen resultaat geeft.
- De app blijft bruikbaar als de externe service tijdelijk niet reageert.
- OpenStreetMap-attributie is zichtbaar op de kaart.

Uitbreidingspunten

- Gebruik `postal_code + name + country_code` om dubbelzinnige gemeenten beter te onderscheiden.
- Toon in de pop-up ook de namen van de collega's uit die gemeente.
- Gebruik aangepaste marker-iconen of clustering.
- Voeg validatie toe zodat foute of onvolledige gemeenten niet opgeslagen worden.
- Voorzie een herbruikbare PHP-functie of serviceklasse voor geocoding.
- Bouw een admin-overzicht van alle gekende woonplaatsen.

Uitbreiding zoeken op naam, voornaam en klikbare hobby's

Korte opdrachtformulering

Maak een zoekfunctie op de publieke collega-pagina waarmee bezoekers kunnen zoeken op voornaam of familienaam. Toon daarnaast hobby's als klikbare badges. Wanneer een gebruiker op een hobby klikt, moet een aparte pagina verschijnen met alle collega's die deze hobby hebben. Stem voor de opslag en opvraging van hobby's af met de leerling die de hobby's relationeel uitwerkt.

Doel

Breid het bestaande smoelenboek uit met een zoekfunctie op voornaam en familienaam. Toon daarnaast hobby's als klikbare badges. Wanneer een gebruiker op een hobby klikt, verschijnt een overzichtspagina met alle collega's die die hobby hebben. Voor deze uitbreiding stem je af met de leerling die de hobby's relationeel uitwerkt.

Databankaanpassingen

Voer minstens deze wijzigingen of afspraken uit:

- ~~Werk met een aparte tabel voor hobby's in plaats van enkel een tekstveld in `smoelboek_colleagues`.~~
- ~~Voorzie een koppeltabel tussen collega's en hobby's.~~
- ~~Spreek met de andere leerling af welke tabelnamen en veldnamen gebruikt worden.~~
- ~~Spreek af of een hobby in links en filters wordt doorgegeven via `id`, `naam` of `slug`.~~
- ~~Gebruik in je eigen code dezelfde databankstructuur als die van de hobby-uitbreiding.~~

Functionele eisen

De uitbreiding moet minstens dit gedrag hebben:

- ~~Er is een zoekveld op de publieke collega-pagina.~~
- De bezoeker kan zoeken op `voornaam` en `familienaam`.
- De zoekterm blijft zichtbaar na het zoeken.
- Alleen collega's die overeenkomen met de zoekopdracht worden getoond.
- Ook als er geen resultaten zijn, verschijnt een duidelijke melding.
- ~~Hobby's worden als losse badges weergegeven.~~
- ~~Elke hobby-badge is klikbaar.~~
- Een klik op een hobby opent een aparte pagina met alle collega's die deze hobby hebben.
- Op die pagina is zichtbaar op welke hobby gefilterd wordt.
- ~~Er is een link terug naar het algemene overzicht.~~

Minimumvereisten

Een oplossing is pas voldoende als minstens aan dit alles voldaan is:

- ~~De zoekfunctie werkt via een formulier met `GET`.~~
- De zoekquery zoekt correct in zowel `first_name` als `last_name`.
- ~~De SQL-query gebruikt prepared statements.~~
- De zoekterm blijft zichtbaar in het formulier na het verzenden.
- De pagina toont een duidelijke melding bij geen resultaten.
- ~~Hobby's worden als klikbare badges weergegeven.~~
- De hobby-overzichtspagina toont correcte collega's op basis van de gekozen hobby.
- De uitvoer van zoektermen en hobbynamen wordt veilig getoond.
- ~~De leerling heeft aantoonbaar afgestemd met de leerling van de hobby-uitbreiding.~~

Uitbreidingspunten

- Toon het aantal gevonden collega's bij een zoekopdracht.
- Maak het mogelijk om ook op woonplaats te zoeken.
- Combineer de zoekfunctie met een hobbyfilter.
- Sorteer collega's alfabetisch op de hobby-overzichtspagina.
- Markeer de zoekterm visueel in de resultaten.

Uitbreiding filteren op man, vrouw of woonplaats

Korte opdrachtformulering

Maak een uitbreiding op het bestaande smoelenboek waarbij bezoekers collega's kunnen filteren op `Man`, `Vrouw` of `Woonplaats`. Toon na het kiezen van een filter alleen de passende collega's en zorg dat de gekozen filter zichtbaar blijft op de pagina.

Doel

Breid het bestaande smoelenboek uit met een filterfunctie waarmee bezoekers snel een selectie van collega's kunnen bekijken. De filter moet minstens werken op geslacht (`Man` of `Vrouw`) en op woonplaats.

Databankaanpassingen

Voer minstens deze controles of afspraken uit:

- Gebruik de bestaande velden voor `gender` en `city` of `city_id` als die woonplaatsuitbreiding al werd uitgewerkt.
- Er zijn geen verplichte nieuwe tabellen nodig voor deze uitbreiding.
- Spreek af met de leerling van de woonplaatsuitbreiding of er gefilterd wordt op de tekstuele woonplaats of op een gekoppelde `city_id`.
- Zorg dat je filterquery aansluit op de databankstructuur die in het project gebruikt wordt.

Functionele eisen

De uitbreiding moet minstens dit gedrag hebben:

- Er is een filterformulier op de publieke collega-pagina.
- De gebruiker kan filteren op `Man`, `Vrouw` of een gekozen woonplaats.
- De gekozen filter blijft zichtbaar na het verzenden.
- Alleen collega's die aan de gekozen filter voldoen worden getoond.
- Er is een duidelijke optie om opnieuw alle collega's te tonen.
- Als er geen resultaten zijn, verschijnt een duidelijke melding.

Minimumvereisten

Een oplossing is pas voldoende als minstens aan dit alles voldaan is:

- De filter werkt via een formulier met `GET`.
- De filter op geslacht werkt correct voor `Man` en `Vrouw`.
- De filter op woonplaats werkt correct voor minstens één gekozen woonplaats.
- De query gebruikt veilige prepared statements als filterwaarden in SQL worden gebruikt.
- De gekozen filter blijft zichtbaar in het formulier na het verzenden.
- De pagina toont een duidelijke melding bij geen resultaten.

Uitbreidingspunten

- Maak het mogelijk om meerdere filters te combineren.
- Toon het aantal gevonden collega's bij de actieve filter.
- Voeg een resetknop toe om alle filters in één keer leeg te maken.
- Combineer de filterfunctie met de zoekfunctie.
- Sorteer de gefilterde resultaten alfabetisch.

Uitbreiding hobby's relationeel uitwerken via Google Form en CSV-import

Korte opdrachtformulering

Maak een uitbreiding op het bestaande smoelenboek waarbij hobby's niet langer als vrije tekst in één veld worden opgeslagen, maar relationeel in aparte tabellen beheerd worden. Verzamel hobbygegevens van collega's via een Google Form, exporteer de antwoorden als CSV en verwerk die via een script in de applicatie.

Doel

Breid het bestaande smoelenboek uit zodat hobby's correct en herbruikbaar worden opgeslagen. In plaats van hobby's als één tekstveld per collega te bewaren, werk je met een aparte hobbytabel en een koppeltabel. Daarnaast ontwerp je een werkbare importflow waarbij collega's hun hobby's doorgeven via een Google Form en de resultaten via CSV in de applicatie verwerkt worden.

Databankaanpassingen

Voer minstens deze wijzigingen en afspraken uit:

- Maak een aparte tabel aan voor hobby's, bijvoorbeeld `smoelboek_hobbies`.
- Voorzie daarin minstens een `id`, een `name` en eventueel een `slug`.
- Maak een koppeltabel aan tussen collega's en hobby's, bijvoorbeeld `smoelboek_colleague_hobby`.
- Leg daarin minstens de relatie tussen `colleague_id` en `hobby_id` vast.
- Spreek met de leerling van de uitbreiding met klikbare hobby's af welke tabelnamen, veldnamen en sleutels gebruikt worden.
- Zorg dat het oude tekstveld met hobby's niet langer de hoofdbron van data is.

Functionele eisen

De uitbreiding moet minstens dit gedrag hebben:

- Een collega kan aan één of meerdere hobby's gekoppeld worden.
- Eenzelfde hobby kan aan meerdere collega's gekoppeld zijn.
- Hobby's worden niet dubbel aangemaakt als ze al bestaan.
- Er is een Google Form waarmee collega's hun naam en hobby's kunnen doorgeven.
- De antwoorden uit dat formulier kunnen als CSV geëxporteerd worden.
- Er is een script dat dit CSV-bestand inleest en de hobby's verwerkt in de databank.
- De gegevensstructuur is bruikbaar voor de andere uitbreiding waarin hobby's als klikbare badges en overzichtspagina's getoond worden.

Minimumvereisten

Een oplossing is pas voldoende als minstens aan dit alles voldaan is:

- Hobby's zitten in een aparte tabel.
- Er is een werkende koppeltabel tussen collega's en hobby's.
- Minstens één collega kan aan meerdere hobby's gekoppeld worden.
- Het CSV-script leest hobbygegevens in en koppelt ze correct aan bestaande collega's.
- Nieuwe hobby's worden aangemaakt als ze nog niet bestaan, bestaande hobby's worden hergebruikt.
- De oplossing vermijdt dubbele hobbyrecords zoals twee keer `voetbal` of `koken`.
- De databankstructuur is bruikbaar voor de uitbreiding met klikbare hobby's.
- Er is aantoonbare afstemming gebeurd met de leerling van de andere hobby-uitbreiding.

Uitbreidingspunten

- Werk met een unieke `slug` per hobby voor gebruik in URL’s.
- Voorzie foutafhandeling voor collega's die niet teruggevonden worden in het CSV-bestand.
- Log welke rijen uit de CSV succesvol verwerkt zijn en welke niet.
- Voorzie een apart beheerscherm om hobby's manueel toe te voegen of te corrigeren.
- Maak het script herbruikbaar zodat later nieuwe CSV-bestanden opnieuw geïmporteerd kunnen worden.