Coding Assignment: Full-Stack Development (.NET Core & Angular)

Hoi kandidaat, dit is de opdracht voor de functie van Full-Stack Developer bij Invenna. Leuk dat je hiermee aan de slag gaat!

Deze opdracht geeft je een indruk van de werkzaamheden waarmee je als Full-Stack Developer bij Invenna te maken krijgt. Tegelijkertijd biedt het jou de kans om te laten zien hoe je te werk gaat en welke vaardigheden je inzet bij dit type taken.

We hopen dat je dit een leuke uitdaging vindt. Voordat je begint, houd alsjeblieft rekening met de volgende richtlijnen en weet dat als je hiermee doorgaat, wij verwachten dat je hiermee akkoord gaat:

- Het opzoeken van online bronnen om je te helpen bij het uitwerken van oplossingen wordt aangemoedigd, maar het is niet de bedoeling om bestaand werk van anderen te kopiëren.
- We vragen je om de details van deze opdracht vertrouwelijk te houden en de opdracht zelfstandig uit te voeren, zonder hulp van buitenaf.
- De code die je schrijft, blijft jouw eigendom, maar wij gebruiken deze uitsluitend voor evaluatiedoeleinden. Jouw werk wordt alleen gedeeld met de betrokkenen in de sollicitatieprocedure.

Opdracht:

Bouw een full-stack applicatie met een .NET Core Web API backend en een Angular frontend voor het beheren van geografische gegevens.

Doel:

Een volledige applicatie ontwikkelen waarbij gebruikers geografische gegevens kunnen importeren, beheren en visualiseren via een intuïtieve gebruikersinterface.

Technical Requirements:

Backend:

- .NET 7.0 of hoger
- ASP.NET Core Web API
- Entity Framework Core
- MS SQL database (SQL Express)
- Exceptionhandling en logging
- Swagger voor API-documentatie

Frontend:

- Angular 17 of hoger
- TypeScript
- Responsive design

Algemeen:

- Git voor versiebeheer
- Duidelijke instructies voor het opzetten van beide delen

Optional Requirements:

- Gebruik van API-versiebeheer
- Unit testing (Backend: xUnit/NUnit, Frontend: Jasmine/Karma)
- Docker-ondersteuning

API Endpoints:

- GET /api/geographicaldata:
    - Geeft een lijst van alle geografische gegevens.
- GET /api/geographicaldata/{id}:
    - Geeft de geografische gegevens met het opgegeven id.

- POST /api/geographicaldata:
    - Maakt een nieuw geografisch item aan.
- PUT /api/geographicaldata/{id}:
    - Werkt bestaande geografische gegevens bij.
- DELETE /api/geographicaldata/{id}:
    - Verwijdert de geografische gegevens met het opgegeven id.

Frontend Features:

- Een dashboard voor het weergeven van geografische gegevens
- Formulier voor het toevoegen/bewerken van geografische items
- Lijst/tabel weergave met sorteer- en filterfunctionaliteit

Geographical Data Model:

- Id (int)
- openbareruimte: string
- huisnummer: int
- huisletter: string
- huisnummertoevoeging: int
- postcode: string
- woonplaats: string
- gemeente: string
- provincie: string
- nummeraanduiding: string
- verblijfsobjectgebruiksdoel: string
- oppervlakteverblijfsobject: int
- verblijfsobjectstatus: string
- object_id: string
- object_type: string
- nevenadres: string
- pandid: string
- pandstatus: string
- pandbouwjaar: int
- x: int
- y: int
- lon: double
- lat: double

Deliverables:

- De broncode van de volledige applicatie (backend en frontend)
- Een README.md met instructies voor het installeren en uitvoeren van de applicatie
- API-documentatie via Swagger
- Database generation met Entity Framework

Evaluatiecriteria:

- Kwaliteit en indeling/structuur van de code
- API-ontwerp en -architectuur
- Frontend-architectuur en componentontwerp
- Gebruiksvriendelijkheid van de UI
- Integratie tussen frontend en backend
- Uitzonderingsbehandeling en foutafhandeling

Optionele evaluatiecriteria:

- Gebruik van API-versiebeheer
- Testdekking en kwaliteit

Opmerking:

- Het CSV-bestand met voorbeelddata wordt door ons geleverd.
- De bovenstaande punten zijn slechts richtlijnen. Je mag extra functionaliteiten toevoegen of uitbreiden waar nodig.