.NET/C# API case

In deze opdracht kan jij laten zien wat jij voor toffe dingen kan bouwen met .NET/C#. Geef deze opdracht vorm zoals jou het beste lijkt en maak hem zo netjes als je maar kan.

Voorwaarden

- Deze case draait om het bouwen van een API; Je hoeft dus geen front-end te ontwikkelen
- Gebruik .NET 5.0 of hoger
- Resultaat aanleveren als GIT repository (bij voorkeur via Github, zorg dat deze door ons gecloned kan worden)
- Zorg voor een readme met instructies voor het opstarten en gebruiken van je project. Benoem hierin ook onderdelen waar je trots op bent en onderdelen waar je misschien minder tevreden over bent en daarbij waarom je trots/minder tevreden bent.
- Lever je endpoint documentatie aan door middel van een Swagger implementatie in je project

Deel 1: API algemeen

Genereer een basis .NET Core Web API. De API moet de volgende functies bevatten:

- Zorg dat de API gebruik maakt van een simpele SQLite database (zorg dat het bijbehorende .db bestand in je project staat)
- Maak een database model voor adresgegevens (straat, huisnummer, postcode, plaats, land)
- Maak een controller met de volgende endpoints:
    - **GET /addresses** Voor het ophalen van meerdere adressen
    - **GET /addresses/:id** Voor het ophalen van een enkel adres
    - **POST /addresses** Voor het toevoegen van een nieuw adres in de database
    - **PUT /addresses/:id** Voor het bewerken van een adres in de database
    - **DELETE /addresses/:id** Voor het verwijderen van een adres uit de database
- Zorg dat bij het aanmaken en wijzigen van je adres alle velden verplicht zijn en hier op gevalideerd wordt

Deel 2: Filters

Voeg aan het **GET /addresses** endpoint functionaliteit toe zodat je het resultaat kan doorzoeken en sorteren en zorg dat deze opties toe te passen zijn door query parameters.

- Zoeken moet kunnen door een zoekwaarde op te geven die alle adresvelden doorzoekt op matches
- Sorteren moet kunnen op alle adresvelden in oplopende en aflopende volgorde

TIP: Probeer dit zo netjes (en generiek) mogelijk te implementeren. Probeer het gebruik van een grote LINQ query / switch / if statement voor het filteren en sorteren te voorkomen. Zoek naar een manier om dynamisch velden uit het model te doorzoeken, zodat wanneer er ooit een adres veld bij zou komen, je geen extra code voor het filteren en sorteren hoeft toe te voegen. Uiteraard kan je hier een package voor gebruiken, maar uiteindelijk zien we liever hoe je dit met eigen code zou oplossen :)

Deel 3: Afstanden

Verbind met een publieke geolocation API, die het mogelijk maakt om afstanden te berekenen tussen twee adressen uit de eerder gemaakte database. Maak hiervoor een extra endpoint waarmee de afstand van een adres A tot een ander adres B opgehaald kan worden (in kilometers). Geef dit je eigen invulling om dit zo netjes mogelijk te implementeren.

Kom je er niet uit, of heb je te weinig tijd om alle delen te implementeren? Geen probleem. Geef dan tekstueel aan in je readme hoe je het zou doen, en licht daarbij je denkwijze goed toe.