Datafication

We have been summoned as the keynote speaker of the annual EF Core seminar. This is a fantastic opportunity to demonstrate our excellence with the EF Core Framework. As a little practice before the demonstration, we want to play around using the EF Core.

Assignment description

Below all requirements for this assignment are listed:

Initial setup

- Create an empty database using SQLite Browser
- Create a database context called **IceCreamDbContext** which should inherit **DbContext**
- Setup all entities and **DbContext** configurations
- Create migrations and update the database
- Run the population script
- Store connection string in config
- Register database context for DI

Entities

- **IceCream**
    - Id
    - Name
    - Description
    - ManufacturerId
    - CreatedDate
    - ModifiedDate

- **Image**
    - Id
    - IceCreamId
    - Url

- **Manufacturer**
    - Id
    - Name
    - Bio
    - ExternalUrl
    - CreatedDate
    - ModifiedDate

- IceCreamCategory
    - IceCreamId
    - CategoryId

- **Category**
    - Id
    - Name
    - ParentCategoryId
    - CreatedDate
    - ModifiedDate

Database queries

For all database read queries all the fields should be populated in the data-transfer object

Read queries

- **IceCreamRepository**
  - GetAllIceCreams => IEnumerable<IceCreamDto>
    - Returns a list of all ice creams in the database
  - GetIceCreamById => IceCreamDetailsDto
    - Returns a single ice cream detailed element
- **ImageRepository**
  - GetImageById => ImageDetailsDto
    - Returns a single image detailed element
  - GetAllImagesByIceCreamId => IEnumerable<ImageDto>
    - Returns a list of all images linked to a specific ice cream
- **ManufacturerRepository**
  - GetManufacturerById => ManufacturerDetailsDto
    - Returns a single manufacturer detailed element
- **CategoryRepository**
  - GetIceCreamsByCategoryId => IEnumerable<IceCreamDto>
    - Returns a list of all ice creams linked to a specific category and if the category has a parent category, the ice creams linked to that category should be provided as well. The chain should not be recursive and should only look for the next in line, but stop after that

Write queries

- **IceCreamRepository**
  - CreateNewIceCream
    - Should create a new ice cream and return a created status code and the location header set on the URL to retrieve the created resource
  - UpdateIceCream
    - Should update all provided fields according to the input model
  - DeleteIceCream
    - Should delete the ice cream based on the id
  - AddIceCreamToCategory
    - Should link an ice cream and category together if they both exist in the database, beforehand
- **ImageRepository**
  - CreateNewImage
- **CategoryRepository**
  - CreateNewCategory
  - DeleteCategory