Assessment - Brewery and wholesale management

Welcome to Belgium! You have been contacted to create a management system for breweries and wholesalers.
Below are listed the functional and technical requirements sent by your client.

Functional requirements

FR1- List all the beers by brewery.

FR2- A brewer can add new beer.

FR3- A brewer can delete a beer.

FR4- Add the sale of an existing beer to an existing wholesaler.

FR5- A wholesaler can update the remaining quantity of a beer in his stock.

FR6- A client can request a quote from a wholesaler, 2 scenarios:

1- If successful, the method returns a price and a summary of the quote.
- o A 10% discount is applied above 10 drinks
- o A 20% discount is applied above 20 drinks

2- If there is an error, it returns an exception and a message to explain the reason:
- o The order cannot be empty
- o The wholesaler must exist
- o There can't be any duplicate in the order
- o The number of beers ordered cannot be greater than the wholesaler's stock
- o The beer must be sold by the wholesaler

Business constraints

- A brewer brews one or several beers.
- A beer is always linked to a brewer.
- A beer can be sold by several wholesalers.
- A wholesaler sells a defined list of beers, from any brewer, and has only a limited stock of those beers.
  The beers sold by the wholesaler have a fixed price, imposed by the brewery.
- For this assessment, it is considered that all sales are made without VAT.
- The database is pre-filled by you.

Data examples

- Brewery
  - Name: Abbaye de Leffe
- Beer
  - Name: Leffe Blonde
  - Alcohol content: 6,6 %

    - Price: 2,20
- Wholesaler
    - Name: GeneDrinks
- Wholesaler's stock
    o GeneDrinks has 10 Leffe Blonde in stock.

Deliverables

- No frontend is needed. You must create an API which should respond to HTTP requests.
- Use REST architecture
- Use the most suitable architecture in terms of maintainability and considering the context of the project.
  The main focus should be put on the architecture, design patterns and C#/Java language features.
- Use Entity Framework (6 or Core) or Hibernate
- Add some unit tests for your code (just to see how you did it).
- **The code should be pushed to a Git repository.**
  Please commit frequently so we can see your progress.

If you have any questions about this assessment, feel free to contact Benoit: benoit@iterates.be

You can send the Git repository link when you are done: hr@iterates.be

When you finish and send the test, please book right away a slot for the technical interview using this link:

https://cal.com/benoit-daccache/technical-interview