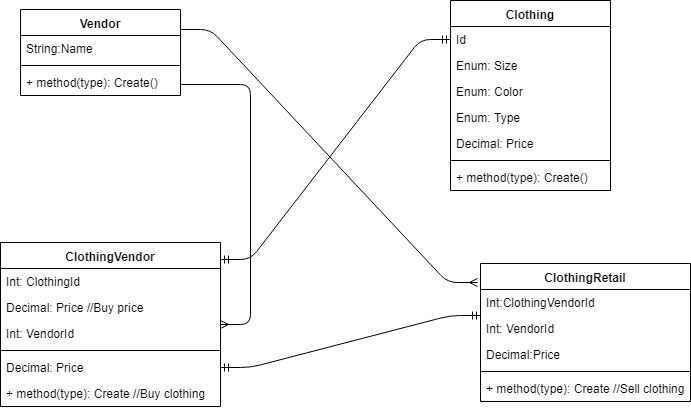
# Solution

I decided to use Net Core API with the help of Swagger to documents functions as well as simulate the calls to execute functions to input data, it is more visualize than using console application.

Below is question from you:

* How would the app scale to support new kinds of clothing other than shirts?
  + All closing’s properties will be dynamically load from database (ex: ClothingType, Color, Size…) Due to time limited, I just define enumerate classes for those properties.
* How would it handle ordering from different suppliers?
  + ClothingVendor table now holds the ordering from multi suppliers, identified by VendorId and ClothingId (see below). For the real product, we should think about multi-tenant pattern.
* What other parts of the app might need to scale?
  + In term of data storage, the ClothingVendor and ClothingRetail will get bigger and need to scale in the future.
  + In term of client performance, for those kind of systems, the most common part usually need to be scaled when product goes-live is view item flow (viewing clothing’s specification, images…). It relates to the read data operation.
  + For searching, We can apply an Elastic Search server.

# Class diagram



# Pre-require:

* Visual studio 2017
* Net core 2.1

# Setup and Run:

## Setup:

* Go to appSetting.json, turn on UseInMemoryDatabase if you want to run on Memory mode (default is Sqlite)
* Delete AltSource.db in root folder and run below command in Package Manager Console if you want to create a new db file. (Select AltSource.Entity as default project)

Add-Migration InitialCreate

Update-Database

## Build and Run:

* Restore packages and build the app.
* Launch app in Debug mode
* Click “Swagger API specification” button to go to Swagger (<http://localhost:44341/swagger/index.html>)

1. Create Clothing items
2. User swagger GUI console to create new Clothing item by calling PUT /api/Clothing
3. Check if data is available by calling Get /api/Clothing.
4. Create Vendor items
5. User swagger GUI console to create new Vendor item by calling PUT /api/Vendor
6. Check if data is available by calling Get /api/Vendor
7. Vendor buy a clothing:
8. User swagger GUI console to create new ClothingVendor by calling PUT /api/ClothingVendor
9. Check if data is available by calling Get /api/ ClothingVendor
10. Vendor by a clothing:
11. User swagger GUI console to create new ClothingRetail by calling PUT /api/ClothingRetail
12. Check if data is available by calling Get /api/ ClothingRetail

# Run on Web (Angular)

(This part is added after test time)

* Download source code for web-base on <https://github.com/neitcouq/altsource-angular>
* Install Nodejs, Anguar CLI
* Go to environment.ts to change backend URL if your server rusn on other endpoint than http://localhost:44341
* Run: npm install to restore packages
* Run: ng serve to launch the web.
* Access the web via default url: <http://localhost:4200>