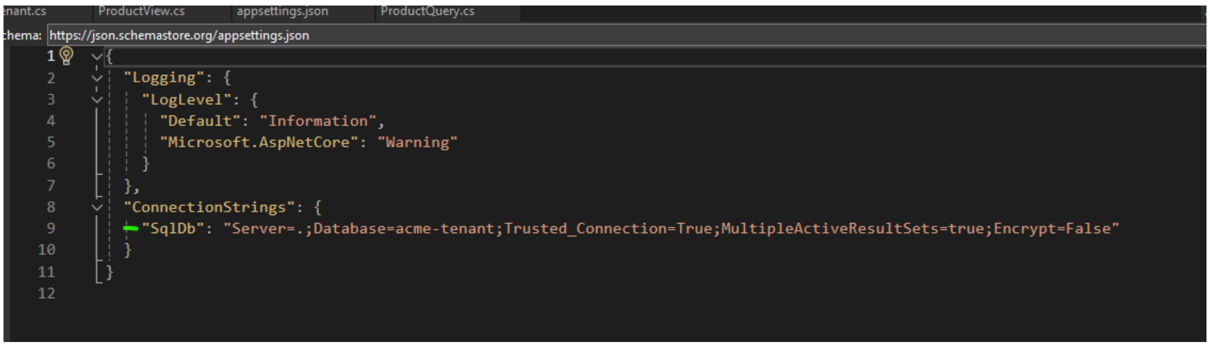
Start the project

Setup the connection string on “appsettings.Development.json”:



```json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "ConnectionStrings": {
    "SqlDb": "Server=.;Database=acme-tenant;Trusted_Connection=True;MultipleActiveResultSets=true;Encrypt=False"
  }
}
```

After setup the connection string, run the project. On the initialization of the project, the migrations will be applied and will be created the follow data:

Tenant:
Name: Acme

Admin User:
Email: admin@email.com
Password: Admin!

Apis

You can follow the api basic info on Swagger, on the address:

[asp.net-core location]/swagger/index.html.

Also, it’s shared the postman collection. Find the file: Api.postman_collection.zip on the main folder of the project.

User management

Create a new user using the endpoint: api/account/signup

To signin and get an authorization token, use the api: api/account/signin.

Tenant management

The Crud operation is available on the endpoints: api/tenant

This operation is only allowed for users from Acme Tenant.

Category management

The Crud operation is available on the endpoints: api/category

Product management

The Crud operation is available on the endpoints: api/product

Project Architecture

The project was divided in the follow layers:

Acme.TechnicalTest.Api

Contain the controllers.

Acme.TechnicalTest.Application

Contain the Use Cases that is the orchestration between the other layers. Create the domain objects, call the repositories and the Unit of Work to save the changes on database.

Also has the classes to handle with domain events handler.

Acme.TechnicalTest.Core

Contain common classes for all the project

Acme.TechnicalTest.Domain

Contain the domains classes of the projects, the contracts and the DTO (data transfer messages), views, message class for the events, and the contracts (interfaces) to be implemented in the other layers.

Acme.TechnicalTest.Infraesctructure

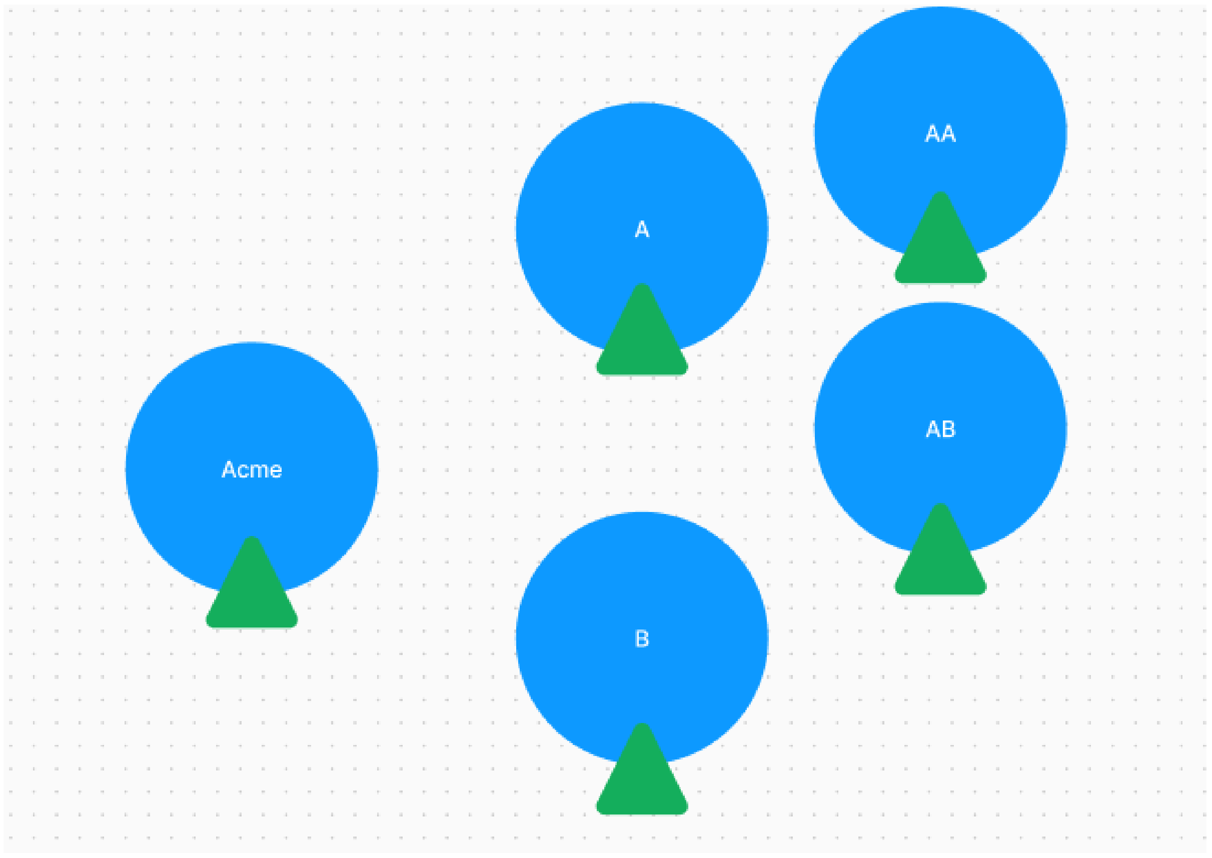
Contain the repositories and queries.

Repositories: Classes to handle with domain operations. The methods return domain objects.

Queries: Classes that return views to be displayed on the pages.

Technical Decisions

Clone data between Tenants



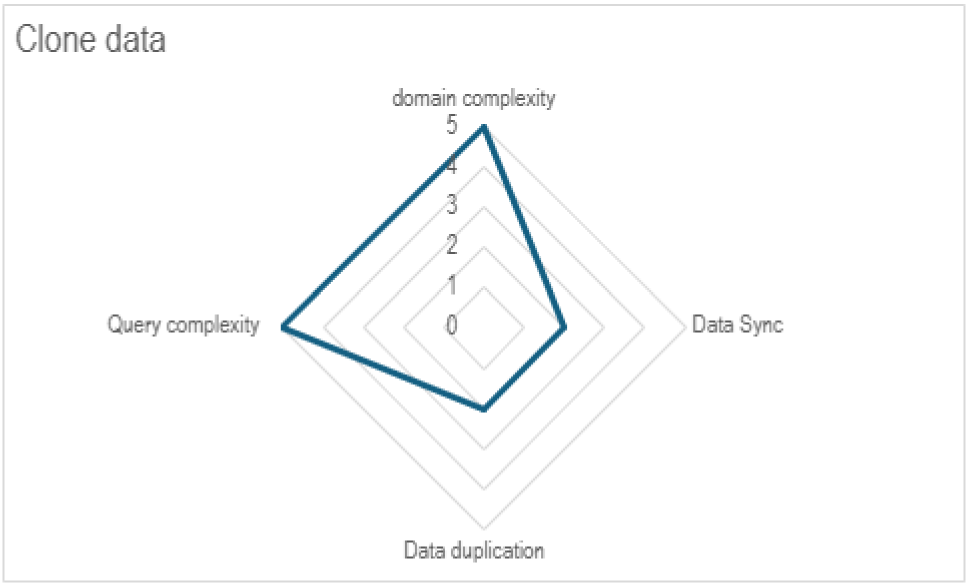
All the products are cloned between the Tenants

To share the data between the Tenants, I used domain events approach using Mediator library. This decision I take was to keep the domain free to handle only with business things and doesn’t need to concern about data distribution.

Also, the query complexity decreases to get the data from the Tenants.

With this approach, the business was successfully attended to.

The drawback of this is to keep the data sync and the record duplication on database. Because when the product is created, this record is cloned in all the children's tenant.



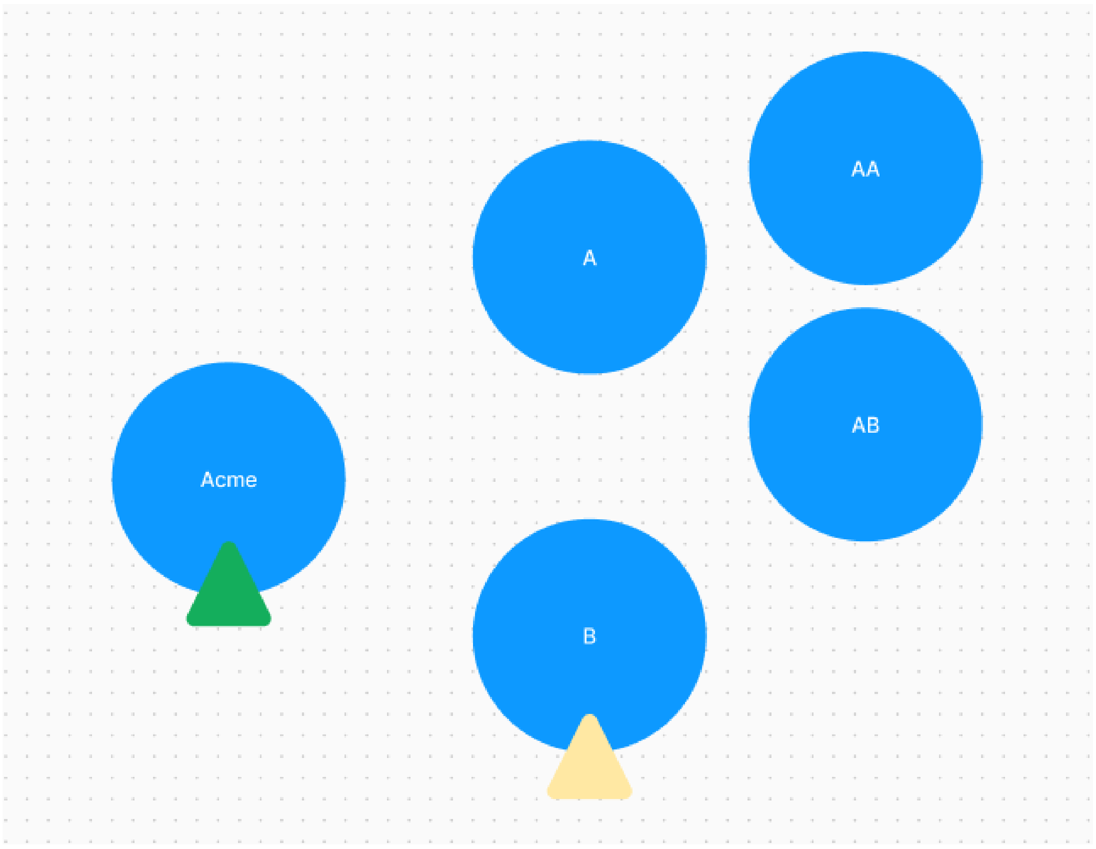
Take in consideration where 5 is less and 1 is high complexity.

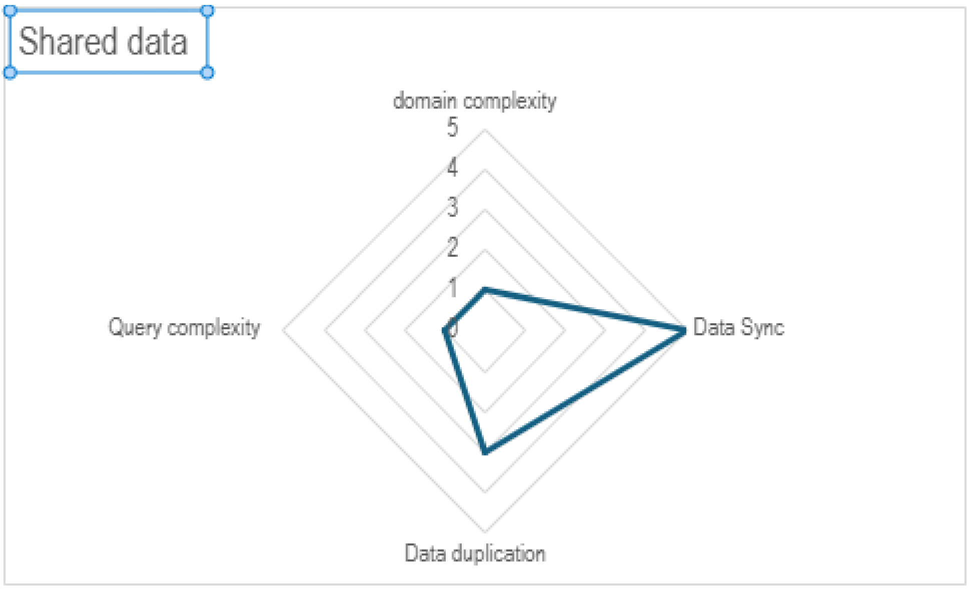
Domain and query complexity have less complexity.
Data sync has high complexity.
Has duplication record on db.

Shared Data, copy only for custom

I haven't created any POC (Prove of concept), so this is just my feelings about this approach.

On this approach the data is only cloned when the Tenant customize the Product. All other Tenants that have the same information get the information from parent tenant.





Take in consideration where 5 is less and 1 is high complexity.
On the graphic the domain and query complexity are too high to keep. In the other hand the data sync is not necessary.

Business doubts

This is some doubts that I had:

About Tenant:
When I delete a tenant (considering data inheritance)

- Should I delete children's tenants?
- If a parent Tenant is deleted, who will be the parent for the orphans Tenant?
- If a child Tenant has customized record, should I delete this record?
- If I have this structure: Acme->A->AA, if I delete the Tenant "A" that is in the middle, how I will handle with the products that AA inheritance from "Acme"?

Product and Category Management

- If a child Tenant create a product with name "Coke" them the parent tenant create a product with the same name. For business purposes, how the system should check if the product is the same or not?
  The same applied for category.
- If is deleted a category with products, how to handle with it? - Currently is throwing a business exception.

Technical Debts

No test was applied

I know that is no excuses for that, but due the lack of time I need to kill the tests. The test on this project is extremally necessary to validate if the domain event is being added and to test event handlers to keep the sync logic working fine.

Delete Tenant Hierarchical Data

Due a lot of doubts, I didn’t do any logic here.

Microsoft Aspnet Identity

This library is deprecated, but I used on this project due the familiarity with the library and my lack of time to find other library with the same purpose.

Event Handlers

The event handlers are done by Hangifire. A queue solution could be a good option here.

You can check the Hangifre panel on the address:

[asp.net-core location]/swagger/hangfire