HassesWebshopCRM Architecture

HassesWebshopCRM followed Domain-driven design (DDD). DDD is the concept that the structure and language of software code (class names, class methods, class variables) should match the business domain.

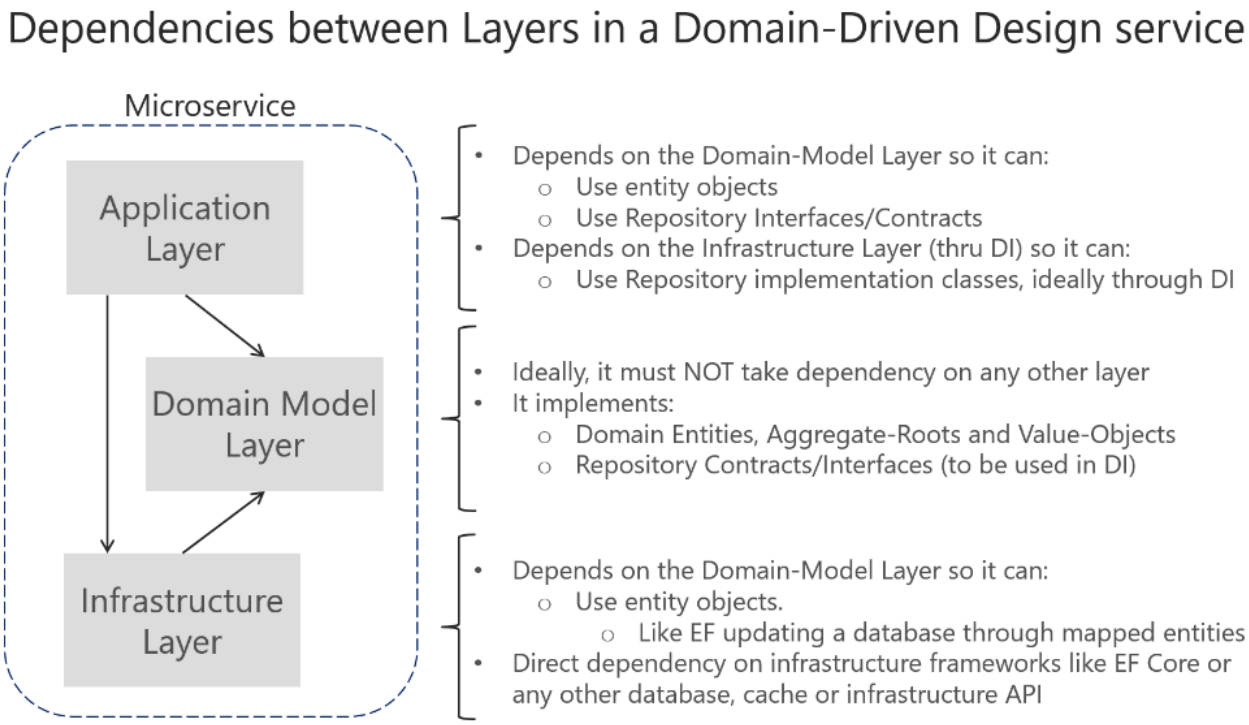

Figure 1: DDD layers

Benefits of DDD:

Eases Communication:

DDD comes in quite helpful when it comes to helping the team creating a common model. The teams from the business' side and from the developer's side can then use this model to communicate about the business requirements, the data entities, and process models.

Provides Flexibility:

DDD turns around the concepts of object-oriented design. This implies that almost everything in the domain model is based on an object and therefore will be modular and encapsulated, enabling the system to be changed and improved regularly and continuously.

Improved Patterns:

Domain-Driven Design gives software developers the principles and patterns to solve tough problems in software and, at times, in business.

Reduced Risk of Misunderstandings:

Requirements are always explained from a domain perspective. Conceptualising the software system in terms of the business domain reduces the risk of misunderstandings between the domain experts and the development team. This also reduces the risk of 'back-and-forth' while freezing requirements.

Better Team Coordination:

The coordination is more people driven (since everyone is using the same terminology) that is less top driven, ensuring fewer bottlenecks.

Here HassesWebshopCRM followed DDD guidelines,

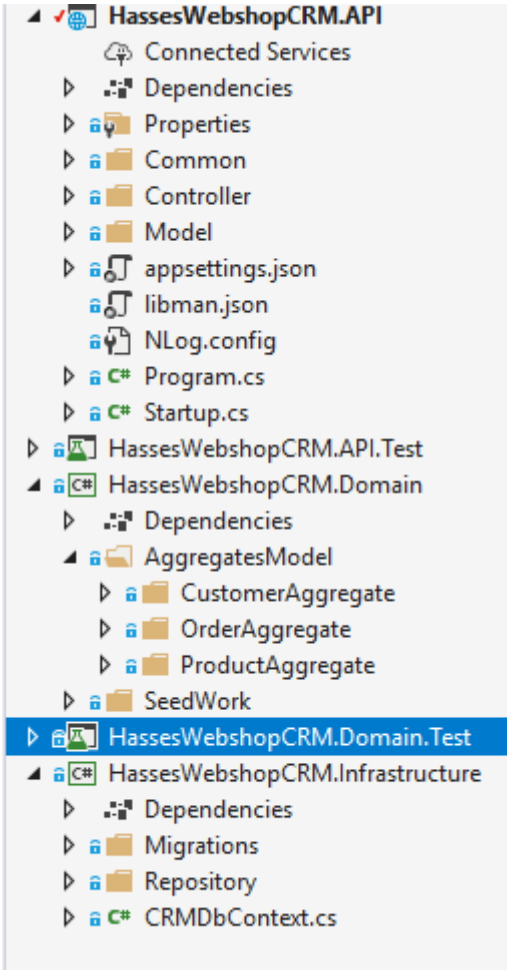

Figure 2: DDD layers in the ordering in HassesWebshopCRM

HassesWebshopCRM Future improvement:

1. Implement Auto Mapper. It will help to auto map between domain and API entity. Creating separate model class product and customer in API project. Currently it uses domain model. It violates DDD rules.

2. Adding more unit test scenario
3. Handling more error exception
4. Create another layer called application layer. It will reduce responsibility from API project layer.
5. Add Integration test.

Using Technology stack:

.Net core web API, Nlog, Entity framework, MOQ, XUNIT, SQL DATABSE