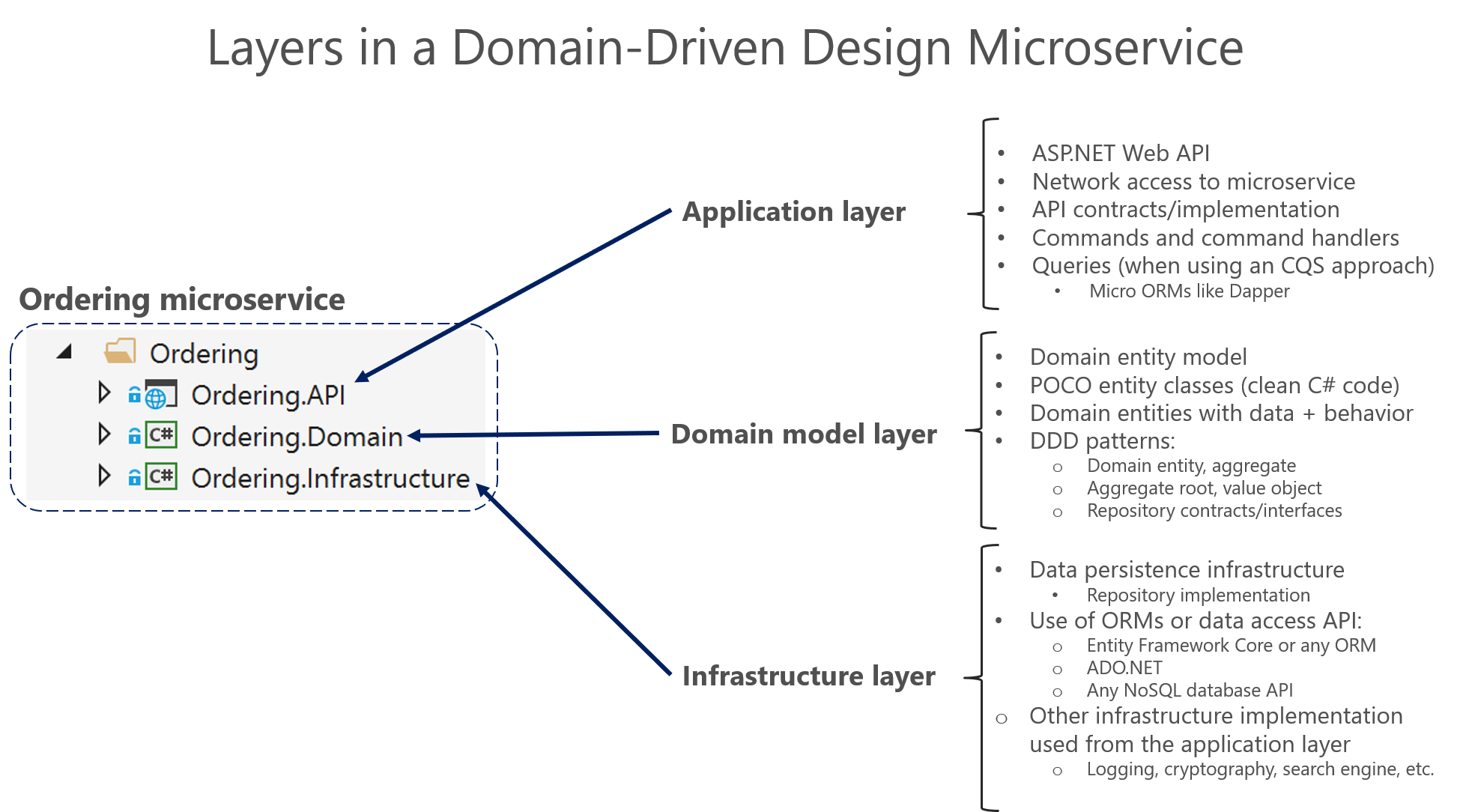
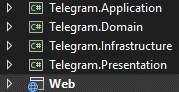
DDD (Domain Driven Design, предметно-ориентированное проектирование)



Также добавим 2 слоя:

* Application
* Presentation

Получим следующее представление:



1. Слой Application необходим для описания команд, хэндлеров и поведений запросов. Реализуется через MediatR.
2. Слой Presentation необходим для описания, например контроллеров. Данный слой будет включать в себя все зависимости других слоев
3. Слой Domain – Core приложения, содержит базовые типы, абстракции, сущности, базовые ошибки и т.д.
4. Слой Infrastructure – содержит в себе реализацию репозиториев и контекст данных

Репозиторий и агрегат

Репозиторий — это очень популярный шаблон для реализации слоя доступа к данным в .NET-приложениях, даже если они не используют шаблоны DDD. Однако, изначально он рассматривался в контексте работы с объектной моделью предметной области именно в контексте DDD, для того чтобы получить из базы данных агрегат и работать с коллекцией агрегатов.

Агрегат — набор объектов предметной области, которые сохраняются в источнике данных (например, реляционной БД), используются совместно в логике приложения, часто имеют связи на уровне базы данных, а также имеют свойство меняться в рамках одной транзакции по бизнес-процессу; агрегат представлен корневой сущностью с ссылками на зависимые объекты. К зависимым объектам мы получаем доступ именно через корень агрегата. Вся бизнес-логика для работы с данными внутри агрегата происходит через корень.

Некоторые NoSQL базы можно вполне считать "агрегатоориентированными", например, документы в Mongo представляют из себя именно агрегатный формат хранения данных, транзакционность гарантируется на уровне одного документа. Более детально эта идея раскрыта в книге "NoSQL Distilled" Мартина Фаулера.

Теперь стоит рассмотреть в чем преимущества использования агрегатов при реализации бизнес-логики.