* 1. Технологические решения, поддерживающие эксплуатационный цикл программы.
     1. Onion-Architecture

Термин "Onion Architecture" ("луковая" архитектура) был предложен Джеффри Палермо (Jeffrey Palermo) еще в 2008 году. Спустя годы данная концепция стала довольно популярной и является одной из наиболее применяемых типов архитектуры при построении приложения на ASP.NET.

Onion-архитектура представляет собой разделение приложения на уровни. При чем есть один независимый уровень, который находится в центре архитектуры. От этого уровня зависит второй уровень, от второго - третий и так далее. То есть получается, что вокруг первого независимого уровня наслаивается второй-зависимый. Вокруг второго наслаивается третий, который также может зависеть и от первого. Образно это может быть выражено в виде лука, в котором также есть сердцевина, вокруг которого наслаиваются все остальные слои, вплоть до шелухи.

Количество уровней может отличаться, но в центре всегда находится модель домена (Domain Model), то есть те классы моделей, которые используются в приложении и объекты которых хранятся в базе данных:

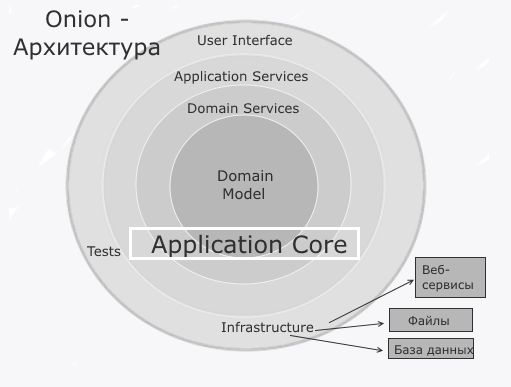


Рисунок – представление onion-архитектуры

Первый уровень вокруг модели домена образуют интерфейсы, которые управляют работой с моделью домена. Обычно это интерфейсы репозиториев, через которые мы взаимодействуем с базой данных.

Внешний уровень представляет такие компоненты, которые очень часто изменяются. Обычно внешний уровень образуют пользовательский интерфейс, тесты, какие-то вспомогательные классы инфраструктуры приложения. К этому уровню также относятся конкретные реализации интерфейсов, объявленных на нижележащих уровнях. Например, реализация интерфейса репозитория, который объявлен на уровне Domain Services. Вообще все внутрение уровни, которые можно объединить в Application Core, определяют только интерфейсы, а конкретная реализация этих интерфейсов располагается на внешнем уровне.

Также стоит отметить, что все внешние хранилища, как базы данных, файлы, внешние веб-сервисы, от которых мы можем получать данные, - все это является внешним по отношению к архитектуре.

При создании архитектуры приложения надо понимать, что реальное количество уровней здесь весьма условно. В зависимости от масштаба задач уровней может быть и больше, и меньше. Однако важно понимать сам принцип, что в центре у нас модели домена, а все остальное зависит от них. Каждый внешний уровень может зависеть от внутреннего, но не наоборот.

В качестве модели данных выступает сущность «Подразделение» с полями:

- Идентификатор объекта (id)

- Название подразделения (Name)

- Состав(Composition)

- Численность (Strength)

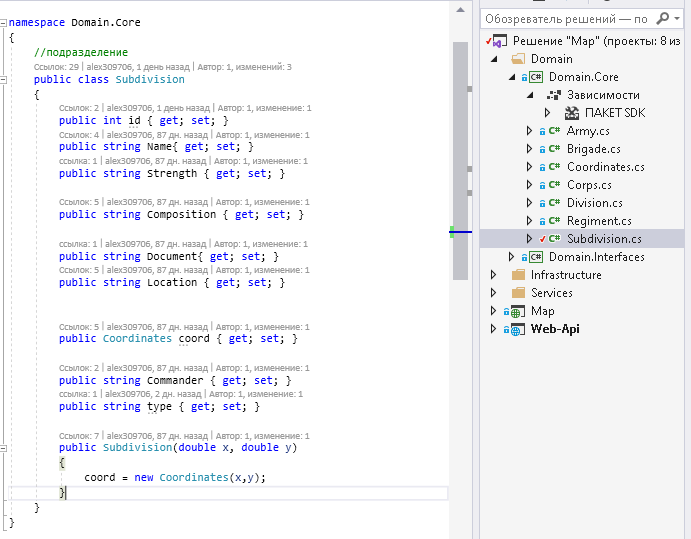
- Удостоверяющий документ(Document)

- Местоположение (Location)

- Координаты на (Location)

- Командующий(Commander)

- Тип подразделения (type)



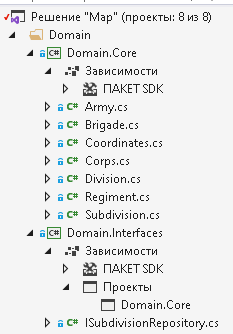


Рисунок – компонент Domain