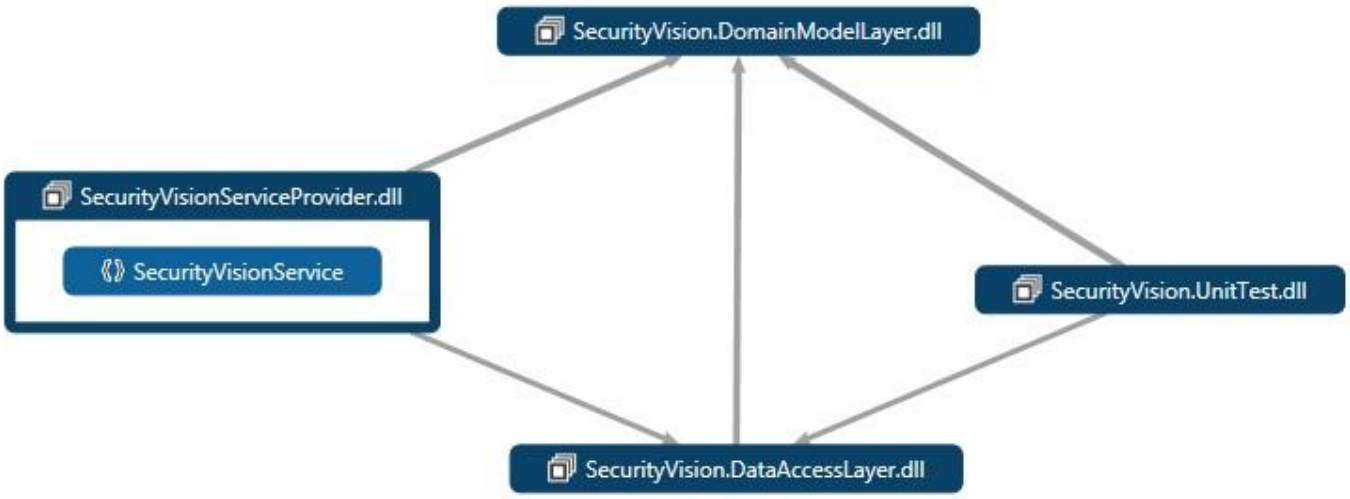

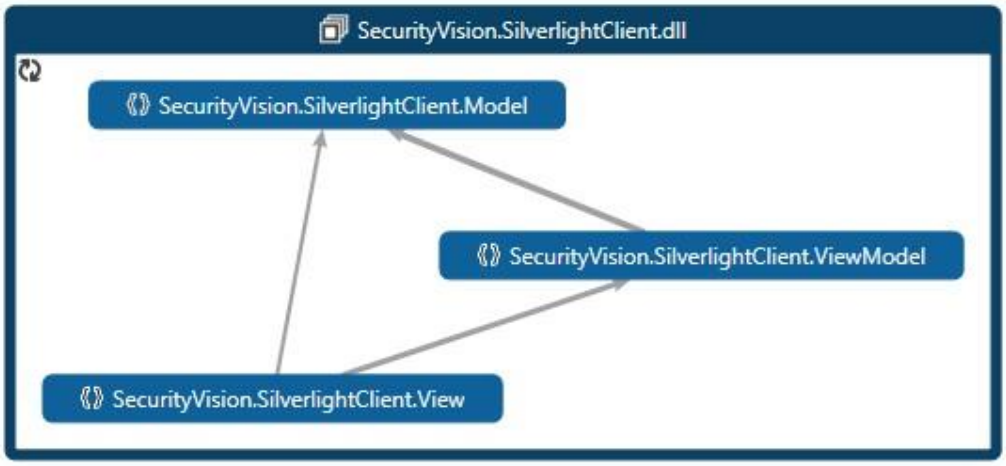

Я реализовал не совсем стандартные подходы на уровне доступа к данным (Code First, вместо привычного мне Database First) и на уровне сервиса (REST, вместо использования стандартного SOAP).

На уровне доступа к данным я реализовал Code-first – подход Entity Framework, что позволило прямо в доменной модели определить атрибутами и контракт для сервиса, и правила для генерации реляционного хранилища, согласно доменной модели.

На уровне сервиса я реализовал архитектурный шаблон REST, что позволило обмениваться с сервисом сообщениями в компактном формате JSON.
Извлечения данных можно получить GET-запросами вроде `http://localhost:50500/SecurityVisionService/Order`, а обновления, удаления и вставки соответствующими POST-запросами, асинхронными, от Silverlight-клиента (view model содержит асинхронные методы обращения к сервису, а view подписывается на события окончания этих обращений).
Подробнее все четыре типа запросов можно увидеть в юнит тестах для слоя сервиса, в проекте: SecurityVision.UnitTests\ServiceLayerTests.
Я подумал, Вам будет любопытно посмотреть это.

На уровне клиента я воспользовался, для отображения фильтров в сетке, компонентами Component One.
Вот любопытная статья, где сравниваются сетки различных производителей и побеждает Telerik, а Component One следует вторым номером.
Клиент реализован, согласно MVVM-паттерну, что можно увидеть на диаграмме зависимостей (ниже в письме). Я использовал привязку к свойствам модели и командам.

Для запуска, первым делом, необходимо создать базу данных, выполнив юнит тест:

SecurityVision.UnitTests\ DataAccessLayerTests\`DataAccessLayer_DatabaseTest()`

При этом, так как произойдёт первое обращении к БД, инфраструктура Entity Framework (code-first) создаст базу **(localDB)\v11.0\SecurityVisionDatabase,**
а юнит-тест заполнит её тестовыми записями.
После этого необходимо запустить решение в режиме отладки (F5), чтобы запустился тестовый веб-сервис, хост для REST-сервиса, по локальному адресу `http://localhost:50500/SecurityVisionService`

После чего можно увидеть работу REST-сервиса, например, командой
`http://localhost:50500/SecurityVisionService/Order` - будут возвращены все заказы в формате json.
Далее достаточно ещё раз выполнить запуск решения, чтобы увидеть клиент, обращающийся с данному REST-сервису.

Далее я кратко описываю решение, а во вложении – сам solution VS 2012.

Решение разбито на слои:

1) доменная модель - проект DomainModelLayer
2) доступ к данным - проект DataAccessLayer
3) сервис - проект ServiceLayer
4) интерфейс – проект SilverlightClient

Также добавлены проекты:

5) юнит тесты - проект UnitTests
6) веб-хост для клиента - проект SilverlightClient.Host

Вот диаграмма зависимостей, показывающая ключевые связи между слоями приложения (в архиве, файл AssemblyDependencies.dgml):