Обоснование решения
Задача: Вариант 3. Книги.

1. Продажа книг.
2. Организация каталога.
3. Использование стопки книг в качестве подставки вместо ножки стола.
4. Поиск информации в книге.
5. Сбор макулатуры.

Комментарии к решению.
В данном проекте представлен пример трехуровневой архитектуры. Трехуровневая архитектура нужна для создания слабосвязанного кода.
В данной задаче это показано на примере класса книги. В слое object-relational-mapping (ORM) представлен класс книги для связи с базой данных. Он автоматически сгенерирован с помощью подхода Code first на основе уже существующей базы данных. В data-access-layer (DAL) представлены шаблоны репозитория и единицы работы. Репозиторий нужен для инкапсуляции запросов к базе данных. Единица работы по завершении транзакции выявляет все изменения и вносит их в БД. В слое business-logic-layer (BLL) представлен шаблон сервиса. Сервис представляет из себя сочетание репозитория и единицы работы. В данном случае сервис так же содержит в себе каталог и функционал для работы с ним. В данной модели сущность каталога должна быть представлена так же, как и сущность книги во всех слоях. Но для упрощения приложения были приняты следующие упрощения: каталог содержит в себе все книги из БД, сохранение каталога не реализовано, чтобы не создавать новую таблицу в БД. В данном примере сервис является некоторой абстракцией. В полном варианте для создания покупки/продажи книг следует создать классы пользователя и магазина, включающие в себя каталог и отдельные сервисы для пользователя и магазина.

Таким образом слой интерфейса знает только про слой бизнес логики, слой бизнес логики знает только про слой доступа к информации, а слой доступа к информации знает только про слой объектно-реляционного отображения. Это обеспечивает большую гибкость и расширяемость приложения. Т.е. впоследствии можно заменить интерфейс, источник данных или правила бизнес логики внося изменения только в один слоев приложения.

При построении иерархии классов применялись отношения композиции. Т.е. каталог содержит в себе список книг и сервис содержит в себе каталог.
Все возможные абстрактные классы, например, для репозиториев и сервисов заменены интерфейсами.

Для разрешения зависимостей предполагается использовать какой-либо IOC контейнер. В данном примере это сделано с помощью Ninject.MVC5.