**1. BookStore.API**

Dieses Projekt ist der Einstiegspunkt Ihrer Anwendung und repräsentiert die **Präsentationsschicht**. Es stellt die Endpunkte zur Verfügung, mit denen Clients interagieren können (z. B. Web, Mobilgeräte oder andere Dienste).

* **Controllers Folder**:
  + **BooksController.cs**: Verarbeitet HTTP-Anfragen, die sich auf Bücher beziehen (z. B. Abrufen einer Liste von Büchern, Abrufen von Buchdetails).
  + **BooksUseCaseController.cs**: Verarbeitet speziell HTTP-Anfragen, die sich auf Anwendungsfälle im Zusammenhang mit Büchern beziehen (z. B. Hinzufügen oder Aktualisieren von Büchern über Anwendungsfälle).
  + **OrderItemsController.cs**: Verarbeitet HTTP-Anfragen im Zusammenhang mit Bestellposten (z. B. Hinzufügen, Entfernen oder Anzeigen von Bestellposten).
  + **OrdersController.cs**: Verarbeitet HTTP-Anfragen, die sich auf Bestellungen beziehen (z. B. Erstellen einer Bestellung, Anzeigen von Bestelldetails).

**2. BookStore.Application**

Dieses Projekt enthält die **Anwendungsschicht**, die für die Implementierung der Geschäftsregeln, anwendungsspezifische Logik und die Interaktion mit der Kern- (Domain-)Schicht verantwortlich ist.

* **DTOs Folder**:
  + Data Transfer Objects (DTOs) sind leichtgewichtige Objekte, die zum Übertragen von Daten zwischen den Schichten verwendet werden, insbesondere zwischen der API- und Anwendungsschicht.
  + **BookAdminDto.cs**: DTO für Buchdaten, speziell im Admin-Kontext verwendet, möglicherweise mit detaillierteren oder bearbeitbaren Informationen.
  + **BookCustomerViewsDto.cs**: DTO zum Anzeigen von Buchdaten für Kunden, angepasst für Webansichten.
  + **BookCustomerViewsMobileDto.cs**: DTO zum Anzeigen von Buchdaten für Kunden, angepasst für mobile Ansichten.
  + **OrderDto.cs**: DTO für Bestellungen, das zum Übertragen von Bestelldaten zwischen den Schichten verwendet wird.
  + **OrderItemDto.cs**: DTO für Bestellposten, das zum Übertragen von Bestellpostendaten zwischen den Schichten verwendet wird.
* **Interfaces Folder**:
  + Enthält Service-Schnittstellen, die die Verträge für die von der API verwendeten Dienste definieren.
  + **IBookService.cs**: Schnittstelle, die Methoden zum Verwalten von Büchern definiert (z. B. Hinzufügen, Aktualisieren, Abrufen).
  + **IOrderItemService.cs**: Schnittstelle zum Verwalten von Bestellposten.
  + **IOrderService.cs**: Schnittstelle zum Verwalten von Bestellungen.
* **Mapping Folder**:
  + Enthält Zuordnungskonfigurationen für die Umwandlung zwischen Entitäten und DTOs mithilfe von Bibliotheken wie AutoMapper.
  + **MappingProfile.cs**: Ein zentraler Ort, an dem alle Zuordnungen zwischen Entitäten und DTOs definiert sind.
* **Services Folder**:
  + Enthält Implementierungen der Schnittstellen im Interfaces-Ordner. Diese Services orchestrieren Operationen, die mehrere Anwendungsfälle oder Repositories umfassen.
  + **BookService.cs**: Implementiert IBookService und behandelt die gesamte geschäftliche Logik im Zusammenhang mit Büchern.
  + **OrderItemService.cs**: Implementiert IOrderItemService und behandelt die gesamte Logik im Zusammenhang mit Bestellposten.
  + **OrderService.cs**: Implementiert IOrderService und verwaltet Bestellungen.
  + **UseCaseBookService.cs**: Ein Beispiel für einen Service, der direkt mit Anwendungsfällen interagiert, möglicherweise für komplexere Operationen, die mehrere Anwendungsfälle umfassen.
* **UseCases Folder**:
  + Anwendungsfälle sind spezifische Geschäftsszenarien oder Operationen, die die Anwendung ausführen muss. Sie kapseln in der Regel eine einzelne Operation oder eine Gruppe verwandter Operationen.
  + **AddBookUseCase.cs**: Enthält die Logik zum Hinzufügen eines neuen Buches.
  + **DeleteBookUseCase.cs**: Verarbeitet die Löschung eines Buches.
  + **GetAllBooksUseCase.cs**: Ruft eine Liste aller Bücher ab.
  + **GetBookByIdUseCase.cs**: Ruft ein Buch nach seiner ID ab.
  + **GetBookSummaryUseCase.cs**: Ruft eine Zusammenfassung der Buchdaten ab, möglicherweise für ein Dashboard oder eine Zusammenfassungsansicht.
  + **UpdateBookUseCase.cs**: Aktualisiert Buchinformationen.

**3. BookStore.Core**

Dieses Projekt repräsentiert die **Domänenschicht** und enthält die grundlegende Geschäftslogik und die Entitäten. Diese Schicht ist unabhängig von allen anderen Schichten und hängt nicht von externen Bibliotheken ab.

* **Entities Folder**:
  + Enthält die Kerndomänen-Entitäten, die die Hauptgeschäftsobjekte darstellen.
  + **Book.cs**: Repräsentiert die Buchentität mit Eigenschaften wie ID, Titel, Autor, Preis usw.
  + **Order.cs**: Repräsentiert die Bestellentität mit Eigenschaften wie ID, Gesamtbetrag, Kunden-ID usw.
  + **OrderItem.cs**: Repräsentiert eine Bestellposten-Entität, die Bücher mit Bestellungen verknüpft.
* **Interfaces Folder**:
  + Enthält Repository-Schnittstellen, die Datenzugriffsoperationen definieren. Diese Schnittstellen werden in der Infrastrukturschicht implementiert.
  + **IBookRepository.cs**: Schnittstelle, die Datenzugriffsmethoden für Bücher definiert.
  + **IGenericRepository.cs**: Eine generische Schnittstelle für allgemeine Datenzugriffsoperationen (z. B. Hinzufügen, Löschen, nach ID suchen).
  + **IOrderItemRepository.cs**: Schnittstelle für Datenzugriffsoperationen im Zusammenhang mit Bestellposten.
  + **IOrderRepository.cs**: Schnittstelle für Datenzugriffsoperationen im Zusammenhang mit Bestellungen.

**4. BookStore.Infrastructure**

Dieses Projekt enthält die **Infrastrukturschicht**, in der die tatsächlichen Implementierungen des Datenzugriffs und der Interaktion mit externen Diensten gespeichert sind.

* **Data Folder**:
  + Enthält Klassen und Konfigurationen, die mit der Datenschicht in Verbindung stehen, einschließlich des Datenbankkontexts und der Repository-Implementierungen.
  + **Repositories Folder**:
    - Enthält konkrete Implementierungen der Repository-Schnittstellen aus der Core-Schicht.
    - **BookRepository.cs**: Implementierung von IBookRepository, die die Persistenz von Buchentitäten behandelt.
    - **GenericRepository.cs**: Implementierung von IGenericRepository, die generische Datenzugriffsmethoden bereitstellt.
    - **OrderItemRepository.cs**: Implementierung von IOrderItemRepository, die die Persistenz von Bestellposten behandelt.
    - **OrderRepository.cs**: Implementierung von IOrderRepository, die Bestelldaten verwaltet.
  + **BookStoreDbContext.cs**: Repräsentiert den Entity Framework DbContext, verwaltet die Verbindung zur Datenbank und die Zuordnung von Entitäten zu Datenbanktabellen.

**Zusammenfassung, wie mit dieser Struktur gearbeitet wird:**

1. **Hinzufügen neuer Funktionalität**:
   * **Schritt 1**: Definieren Sie ein neues DTO im BookStore.Application.DTOs-Ordner, wenn Sie Daten übertragen müssen.
   * **Schritt 2**: Erstellen Sie einen neuen Anwendungsfall im BookStore.Application.UseCases-Ordner, um die Geschäftslogik zu kapseln.
   * **Schritt 3**: Erstellen oder ändern Sie gegebenenfalls einen Service im BookStore.Application.Services-Ordner, um zwischen Anwendungsfällen oder anderen Diensten zu koordinieren.
   * **Schritt 4**: Definieren oder erweitern Sie die Entität im BookStore.Core.Entities-Ordner.
   * **Schritt 5**: Implementieren Sie die Datenzugriffslogik im BookStore.Infrastructure.Data.Repositories-Ordner.
   * **Schritt 6**: Stellen Sie die neue Funktionalität über einen Controller im BookStore.API.Controllers-Ordner zur Verfügung.
2. **Umgang mit Datenzugriff**:
   * Der gesamte Datenzugriff erfolgt über Repository-Schnittstellen, die in der Core-Schicht definiert und in der Infrastrukturschicht implementiert werden. Dies ermöglicht ein einfaches Austauschen der Datenzugriffslogik (z. B. Wechsel von Entity Framework zu Dapper), ohne die Geschäftslogik zu beeinflussen.
3. **Zuordnung zwischen den Schichten**:
   * Verwenden Sie AutoMapper-Konfigurationen im BookStore.Application.Mapping-Ordner, um die Umwandlung zwischen Entitäten und DTOs zu handhaben.
4. **Orchestrierung der Geschäftslogik**:
   * Anwendungsfälle im UseCases-Ordner kapseln einzelne Operationen, wodurch die Geschäftslogik sauber und wartbar bleibt. Dienste im Services-Ordner können komplexe Operationen koordinieren, die mehrere Anwendungsfälle umfassen.