Backend – Aufbau & Struktur  
(klassisches Request-Response Modell)

HTTP-Request => DB – (Web-App =>) Controller => Service => Repository

* WebAPI mit Controller – nimmt HTTP-Requests von der Web-App entgegen  
  XY-Controller bekommt Request von der Web-App, im Controller wird das entstandene Request-Objekt validiert (z.B. auf sinnvolle ID), danach wird das Request-Objekt an den entsprechenden Service übergeben.
* Services – mehrere Services, die für die Übernahme von Request-Objekten aus dem Controller verantwortlich sind und eine Zwischenschicht (Abkapselung des DB-Access!) darstellen.  
  XY Service bekommt vom XY Controller ein Request-Objekt und mappt dieses auf ein entsprechendes Query-Objekt, und übergibt dieses an die (bei uns generische) Repository.
* Repository – liegt „direkt“ an der Datenbank und führt die eigentlichen Abfragen auf dem DbContext aus.  
  Die Repository bekommt vom Service ein Query-Objekt und fragt mit ebendiesem, klassisch CRUD, den DbContext („die Datenbank“) ab und gibt dann ein DTO (data transfer object), was MEHR ODER WENIGER direkt dem Datenbankobjekt entspricht, an den Service zurück.

DB => HTTP-Response – Repository => Service => Controller (=> Web-App)

* Repository  
  Nachdem die DbContext-Abfrage mit dem Query-Objekt abgeschlossen ist, gibt die Repository ein DTO an den entsprechenden Service zurück.
* Service  
  XY Service bekommt von Repository ein DTO und mappt dieses in ein entsprechendes Response-Objekt, welches dann an den entsprechenden Controller übergeben wird.
* WebAPI Controller – übergibt HTTP-Responses an Web-App  
  XY Controller bekommt vom XY Service ein Response-Objekt, macht daraus eine eigentliche HTTP-Response (JSON, Header, Body, StatusCode etc.) und schickt diese an die Web-App