**ASP dotnet Core Notes**

**Resource Naming Guidelines**

* Use nouns not actions. Convey hierarchy via order with children after parents.
* For filters, sorting, and ordering use a query string as they are not resources themselves
* ~~Api/getauthor~~ ☹
* Get api/authors 😊
* ~~Api/id/employees~~ ☹
* Api/employees/{employeeid} 😊
* Guids vs Auto Integer for IDs
  + Using Guids
    - Allow switching of backend data stores without having to map
    - Hides implementation details
    - Guids are not as readable by humans, but there are work arounds

**Importance of Status Codes**

* Indicate whether or not the request worked out as expected
* Report who or what is responsible for a failed request
  + User should know if they messed up
  + Host should know if they messed up
* Codes
  + Level 200
    - 200 – ok
    - 201 – created
    - 204 – No Content
  + Level 400
    - 400 - Bad request
    - 401 – Unauthorized
    - 403 – Forbidden
    - 404 – Not Found
    - 409 – Conflict
  + Level 500
    - 500 – Internal Service Error (uh oh)
* The controller can decide and return IActionResult for all those types.. Ok NotFound etc..

**Formatters and Content Negotiation**

* Selecting the best representation for a response given multiple available representations. Media type is passed via accept header.
  + Output formatter
    - Deals with output media type: accept header
  + Input formatter
    - Media type: content-type header

**EF Core**

* EFC core
  + is a lightweight, x-platform ,and extensible version of EF.
  + is not the same as the original EF
* Annotations are one way to define PK and FK, required fields, validation etc…
  + If you feel comfortable with it, EF has an implied version
* Migrations can be used and applied dynamically, DB can spin up dynamically with seed data
* Seed data can be applied by using an extension method

**The Repository Pattern**

* An abstraction that reduces complexity and aims to make the code, safe for the repository implementation, persistence ignorant.
* Pros
  + Lessens duplication
  + Lessens error-prone code
  + Offers better testability
* Persistence ignorance: switching out the persistence technology is not the main purpose, choosing the best persistence technology (ef, ado, etc…) specific to the need of each repository method is. Allows you to choose a best fit for the current situation while the rest of the code can remain ignorant.

**Automapper**

* [**https://github.com/AutoMapper/AutoMapper**](https://github.com/AutoMapper/AutoMapper)
* Handy for converting models and entities to database objects.
* It’s not necessary, mapping can be done manually, but it does cut down on repetitive code and maintenance. Lots of ways to go about it, your choice. :D