Endpoints

* BulletinEndpoints.MapBulletinEndpoints:
  + Undersøg (fra evt UI) om der er implementeret tilstrækkelig validering af inputparametre som memberId, title og description for at sikre, at de er gyldige og korrekte.
  + Implementer en ILogger
  + Hvorfor egen exception type BulletinExceptions
    - Kan den indholde en http statuskode --> HTTP-statuskode i stedet for Results.Problem for at indikere specifikke fejlårsager.
  + Kan der ikke opstå andre udtagelser BulletinExceptions
* MemberCommentsEndpoints.CreateMemberComment:
  + Do…validering af inputparametre som authorId og body
  + HTTP-statuskode i stedet for Results.Problem
  + Do flere catch --> HTTP-statuskode i stedet for Results.Problem specifikke fejlårsager.
* MemberEndpoint.MapUserEndpoints:
  + Igen validering af inputparametre
  + specifikke HTTP-statuskoder Results.NotFound i stedet for Results.BadRequest for at indikere, at en bruger ikke blev fundet.
  + Er der yderligere undtagelser ud over generiske undtagelser som Exception
  + Måske mere detaljerede fejlbeskeder i HTTP-responsen
* ShareEndpoints.MapBulletinsEndpoints
  + Do

Input skal ske både client og server side

* Brug af datatyper og constraints:
  + Forventede datatype (f.eks. heltal, streng, dato osv.)
  + nødvendige begrænsninger (f.eks. minimumslængde, maksimumsværdi osv.).
  + Begrænsning af interval en numerisk værdi mellem 1 og 100 kontrol
* regex
  + Validere tekstbaserede inputparametre.
  + Mulighed for at definere et mønster, som inputtet skal matche
* eksterne biblioteker <-- avancerede valideringsfunktioner og mulighed for at definere brugerdefinerede regler.
* **public** IActionResult CreateMember**(**string username**,** string email**)**
* **{**
* **if** **(**string**.**IsNullOrEmpty**(**username**))**
* **{**
* ModelState**.**AddModelError**(**"username"**,** "Username is required."**);**
* **}**
* **if** **(**string**.**IsNullOrEmpty**(**email**))**
* **{**
* ModelState**.**AddModelError**(**"email"**,** "Email is required."**);**
* **}**
* **else** **if** **(!**IsValidEmail**(**email**))**
* **{**
* ModelState**.**AddModelError**(**"email"**,** "Invalid email format."**);**
* **}**
* **if** **(!**ModelState**.**IsValid**)**
* **{**
* **return** BadRequest**(**ModelState**);**
* **}**
* // Resten af logikken for oprettelse af medlem
* **}**

Ekception handling

Præcisere catch-blokke:

* I øjeblikket bruges generelle catch-blokke, der fanger alle undtagelser af typen Exception
  + Fang specifikke undtagelser som ArgumentNullException, InvalidOperationException
  + I catch-blokke logges undtagelserne ved hjælp af logger.LogError uden at hele StackTrace logges
* Centraliseret exception-håndtering :)
  + Flot middleware eller en global exception-handler, der fanger undtagelser på tværs af alle endpoints. Dette giver mulighed for mere ensartet og effektiv håndtering af undtagelser i hele applikationen.

Forklar klasserne i Exceptions