

▮ Øvelse 18 – Inputvalidering til konsolapplikation

▮ Information

Formålet med denne øvelse er at anvende jeres viden om **Regex** fra forrige øvelse til at implementere **inputvalidering** i en konsolapplikation.

I denne øvelse, samt den forrige, skal I arbejde med repositoret:

[InputValidationsBasicExercises](#).

Det projekt, I skal fokusere på, hedder `ValidateTheName` og er en konsolapplikation. I filen `Program.cs` finder I metoden `nameInputIsValid`, hvor valideringslogikken skal implementeres.

- ▮ *Husk altid at validere længden af tekststrengen, inden I bruger Regex! En for lang tekststreng kan føre til resourceudtømmning i applikationen.*
- ▮ *En dårlig eller kompleks Regex kan resultere i performanceproblemer og i værste fald føre til Denial of Service (DoS) angreb, hvor applikationen bliver utilgængelig på grund af langvarig behandling. Derfor er det vigtigt altid at kontrollere inputlængden, inden Regex anvendes.*

Hvis du har brug for at genopfriske C#-grundlæggende, kan du kigge på dine noter fra [Opgave 6 – Grundlæggende OOP med C#](#).

▮ Hjælp til:

- [Sådan finder du længden på en tekststreng](#)
- [If-statements i C#](#)
- Test dine Regex-udtryk på [Regex 101](#)

▮ Fokus på konceptet, ikke sproget

▮ Bemærk:

I denne øvelse arbejder vi med *inputvalidering som sikkerhedsprincip*, ikke med C# som sprog.

Det betyder, at vi fokuserer på det **underliggende koncept og sikkerhedsdesign**, som gælder på tværs af programmeringssprog.

Eksemplerne i øvelsen er i C#, men principperne er **generelle og almengyldige**, og kan overføres til mange forskellige sprog eller system, I senere arbejder med.

□ Forskel på inputvalidering og sanitization

Når vi taler om at sikre input i software, er det vigtigt at forstå forskellen på **inputvalidering** og **sanitization**:

- □ **Inputvalidering** handler om at *kontrollere*, om inputtet overholder forventede regler (f.eks. længde, format, tegnsæt osv.). Hvis inputtet ikke er gyldigt, afvises det straks. Det er en *gatekeeper*, som sikrer, at kun korrekt data får adgang.
- □ **Sanitization** handler om at *rette* eller *omforme* inputtet til et sikkert format, som programmet kan håndtere uden risiko. Dette kan f.eks. være at fjerne farlige tegn eller escape input. Men vær opmærksom:
- Sanitization kan utilsigtet "forvride" data, så det mister mening.
- Det kan også åbne op for angreb, hvis en angriber udnytter, at input automatisk ændres til acceptabelt format, f.eks. ved at sende specielle tegn, der bliver transformeret til noget skadeligt.
- Sanitization bør derfor kun bruges som supplement – og ikke som erstatning for grundig inputvalidering.

□ *Husk: Inputvalidering afviser usikre eller forkerte data, mens sanitization renser data, men kan være mindre sikker, hvis den anvendes alene.*

At forstå og adskille disse begreber hjælper jer med at skabe robuste og sikre applikationer, som både beskytter systemet og sikrer dataintegritet.

□ Instruktioner

1. □ Valider, at navnet (`nameInput`) **ikke er længere end 20 tegn**.
2. □ Valider, at navnet **ikke er kortere end 3 tegn**.
3. □ Valider, at navnet **kun indeholder alfabetiske karakterer** ved hjælp af Regex (brug ikke andre metoder til dette).
4. □ Test valideringen ved at køre applikationen og prøve forskellige inputs.
Tip: Sørg for at validere længden, inden du bruger Regex på inputtet.

□ *Tip: Implementer valideringen trin for trin, og test ofte, så I sikrer jer, at hver del virker korrekt, inden I går videre.*

□ Refleksionsspørgsmål

- 1. Hvorfor er det vigtigt at kontrollere længden på input, inden man anvender Regex til validering?
- 2. Hvordan kan dårligt designede Regex-udtryk føre til sikkerhedsproblemer som DoS-angreb?
- 3. Hvilke typer af angreb kan opstå, hvis inputvalidering ikke implementeres korrekt, særligt i API-kontekster?
- 4. Hvordan hænger inputvalidering sammen med konceptet om tillidsgrænser i softwarearkitektur?
- 5. Hvordan kan man kombinere inputvalidering med andre sikkerhedsforanstaltninger for at opnå en mere robust applikation?
- 6. Hvilke overvejelser bør man gøre sig omkring brugeroplevelsen, når man implementerer streng inputvalidering?

Kobling til relevante standarder og principper

Når I arbejder med inputvalidering i denne øvelse, relaterer det sig til flere centrale principper og standarder for software- og informationssikkerhed:

Element i øvelsen	Relateret standard / princip	Kommentar
Inputvalidering	ISO/IEC 27002:2022, afsnit 8.26 (applikationssikkerhed)	Anbefaler validering og integritetstjek af input til en applikation.
	OWASP ASVS 5.0.0, kontrol 2.2. (Input Validation)	Krav om validering af data
	Secure by Design (Johnsson et al.)	Understreger vigtigheden af at beskytte tillidsgrænser og sikre korrekt inputvalidering.

□ *Inputvalidering beskytter systemet ved at sikre, at data, der krydser tillidsgrænser – f.eks. fra brugerinput eller API-kald – er korrekt og sikkert formateret, og dermed mindsker risikoen for injection-angreb og DoS.*

