# Youri van der Star

# Opgeleverde functionaliteit

Tekstbestand selecteren

Cursusplanning importeren

Resultaat bekijken

Ingeplande cursusinstanties bekijken

geen bestaande of dubbele cursusinstanties importeren

Gesorteerde lijst bekijken

geen cursusplanning "in incorrect formaat" importeren

Foutmelding bekijken

Huidige week tonen

Weeknummer (en jaar) kiezen

lijst in gekozen week tonen

weekoverzicht in favorieten

begin- en einddatum opgeven

alleen cursusinstanties binnen periode importeren

Navigeren naar volgende en vorige week

Cursusinstantie kiezen

Cursistgegevens invullen

Aantal ingeschreven cursisten voor elke cursus tonen

* Gesorteerde lijst bekijken ontbreekt. Ik zie in de code ook geen aanwijzingen dat dit gebouwd is

# 

# Kwaliteit

* Import-functie kan niet omgaan met twee witregels aan het eind van het bestand (dit hebben de aangeleverde bestanden)
* Import-functie importeert niet de laatste cursussen in het bestand. De anderen lijken wel geimporteerd te worden
* Backend UploadController.cs bevat veel code in één method. Een aantal andere methods vind ik ook vrij fors in lengte. Beter is om deze op te splitsen in kleinere stukjes logica die één ding doen. Dit maakt de code leesbaarder, beter onderhoudbaar en beter testbaar. Voorbeelden hoe dit korter kan:
  + Inline methods dragen ook bij aan lange methods. Bijvoorbeeld MatchOrThrowInvalidLineException in UploadController.cs zou prima als een losse method kunnen
  + Validatie-logica verplaatsen naar validation attributen (via System.ComponentModel.DataAnnotations)
  + In de backend een laag plaatsen tussen de Controller-laag en de Repository-laag. Deze nieuwe laag handelt dan domein-logica af (checken of cursusinstanties al bestaan, etc.), de controller zorgt dan dat het input-bestand omgezet wordt naar een in-memory object, de repository bevat dan puur logica voor simpele database-acteis (Geef waarde x, schrijf waarde y, update waarde z).
* Paar kleine code smells zoals if-expressies die altijd true of false zijn, naamgeving (\_CursusRepository ipv \_cursusRepository), null-checks op non-nullable types (Duur != null in CursusInstantieBuilder bijvoorbeeld). Hier zouden linting tools of tools zoals ReSharper je kunnen helpen.
* Frontend-code is lekker clean.

# 

# Testbaarheid

* Tip: Kijk eens naar libraries als Shouldly of Fluent assertions. Het gebruik van de standaard asserts is zeker niet fout, maar deze kunnen je asserts leesbaarder maken
* Ik zie gebruik van triple A terug
* Tip: Je hebt Mock-classes gebruikt voor het mocken van je repositories. Prima oplossing, maar kijk ook eens naar frameworks zoals moq. Deze zorgen ervoor dat je zelf minder boilerplate hoeft te schrijven en dat je vanuit je testen al duidelijk aan kan geven wat het gedrag van je gemockte class is

## Front-end

* Mutation score: 66.67
* Code coverage: 67.92%

## Back-end

* Mutation score: 54.78
* Code coverage: 74,31%
* UploadController.cs in zijn geheel niet getest. Zie comment bij kwaliteit. Gezien de hoeveelheid logica in deze class is dit een belangrijke om wel mee te nemen in je test coverage

# 

# Deployment

* Readme kort maar krachtig
* Docker-compose up werkt direct
* Pipelines zien er prima uit
* Kleine tip: Je beide pipelines zijn op wat input parameters na exact hetzelfde. Je zou eens kunnen kijken naar templates zodat je pipeline-code kan sharen ipv dupliceren ( <https://learn.microsoft.com/en-us/azure/devops/pipelines/process/templates> )

# Professionele werkwijze

In het begin was de communcatie van Youri heel strak. Later werden de bijeenkomsten wat rommeliger. Het hielp daarbij niet dat hij had gekozen voor sprints van een halve dag. Zo was er steeds maar weinig te bespreken en had de stakeholder het gevoel dat er veel dingen dubbel zijn gezegd. Het voordeel was wel dat Youri steeds aan het goede werkte. En dat als het verwerken van de feedback niet goed was gegaan (zoals bij het overzicht op de eerste dag) het toch binnen een dag goed kwam.

Uiteindelijk kon Youri de stakeholder wel vertrouwen geven in een goede afloop.

4/5