# Deel 6: Testomgeving en Testdata

In de vorige blog hebben we gekeken naar de voordelen van goede testautomatisering en de risico's van slecht geïmplementeerde testautomatisering en naar hoe je de kwaliteit van testautomatisering bewaakt.

In deze blog kijken we naar de eisen die testautomatisering stelt aan de testomgeving en de testdata.

## Eisen aan de testomgeving en testdata

Een stabiele testomgeving met voldoende en bruikbare testdata is voor handmatig testen een vereiste. Er is niets vervelender dan een test die faalt omdat de testdata incorrect of niet bruikbaar is.

Testautomatisering stelt aanvullende eisen aan een testomgeving. Vanwege het extreem deterministische karakter van geautomatiseerde tests moet de testomgeving en de testdata 100% deterministisch zijn. Het is daarom goed om geautomatiseerde tests op een eigen testomgeving te draaien die niet voor andere doelen (b.v. handmatig testen) of door meerdere teams wordt gebruikt. Veelal zie je dat bij invoering van testautomatisering er een vraag ontstaat naar meer testomgevingen. Technieken als virtualisatie (b.v. Docker, Kubernetes) en Clouddiensten (AWS, Azure, Oracle Cloud, Google Cloud) bieden functionaliteiten die het, on-demand, opspinnen en afbreken van reproduceerbare testomgevingen met reproduceerbare testdata mogelijk maken. Een voorwaarde voor testautomatisering is dat de gebruikers volledige beheerrechten op de testomgevingen hebben.

## Stabiele testdata

In transactionele systemen (b.v. een online shopping systeem) wordt data verwerkt. Een order wordt ingevoerd, betaald en verwerkt tot aan verzending. Door het uitvoeren van een test verandert dus de testdata en is deze niet opnieuw te gebruiken door dezelfde test. Omdat geautomatiseerde tests bij herhaling worden uitgevoerd, moet het dus mogelijk zijn om de testdata voor aanvang van elke test terug te zetten in de begintoestand. Dit kan op twee manieren:

- De testdatabase leeggooien en een kopie van de initiële testdata terugzetten.
- De testdatabase leeggooien en de testdata met behulp van de testautomatisering invoeren voordat de tests beginnen.

Om testen deterministisch te maken moeten externe afhankelijkheden op de testomgeving die niet onder controle zijn zoveel als mogelijk geëlimineerd worden. Dit kan door de echte

instanties (bv een API) te vervangen door een stub of een mock. Een zelf beheerde stub of mock geeft je namelijk de mogelijkheid elke gewenste response te geven die nodig is voor het kunnen uitvoeren van de test. Denk hierbij aan edge cases en foutsituaties (HTTP 40x en 50x responses.

Wanneer gebruik wordt gemaakt van stubs en/of mocks is het belangrijk dat het gedrag ervan gelijk is aan het echte product. Bij invoering van stubs en mocks is het dus van belang om het gedrag eenduidig vast te leggen en te beheren (b.v. in een OpenAPI/Swagger contract).

Testdata en gebruikte stubs en mocks zijn een essentieel onderdeel van testautomatisering en het is daarom belangrijk om ze, net als de code van de geautomatiseerde tests onder versiebeheer te brengen.

## Vooruitblik volgende blog

Tot zover deze blog. In de volgende blog staan we stil bij de rol van testautomatisering in CI/CD.

STAY TUNED!

**Wim ten Tusscher**