Cucumber kombinert med WireMock

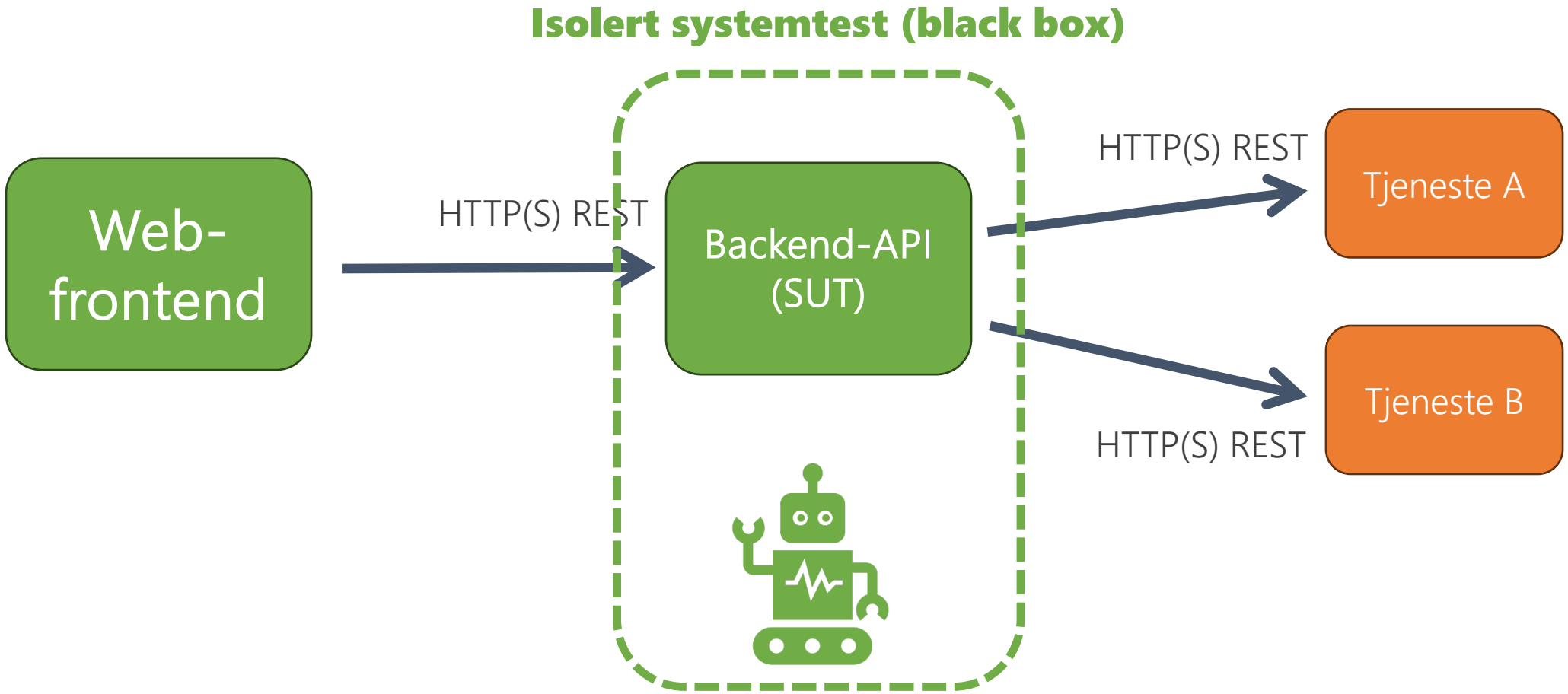
magnus.halvorsen@testify.no

I denne sesjonen:

- Cucumber-scenarier for et REST-API – spesifikasjon, test og levende dokumentasjon i ett?
- WireMock – simulering av et REST-API for å forenkle og forbedre testingen

- Vi tar utgangspunkt i et kjørende eksempel m/kode
- Mulighet til oppgaveløsning for den som vil 💪

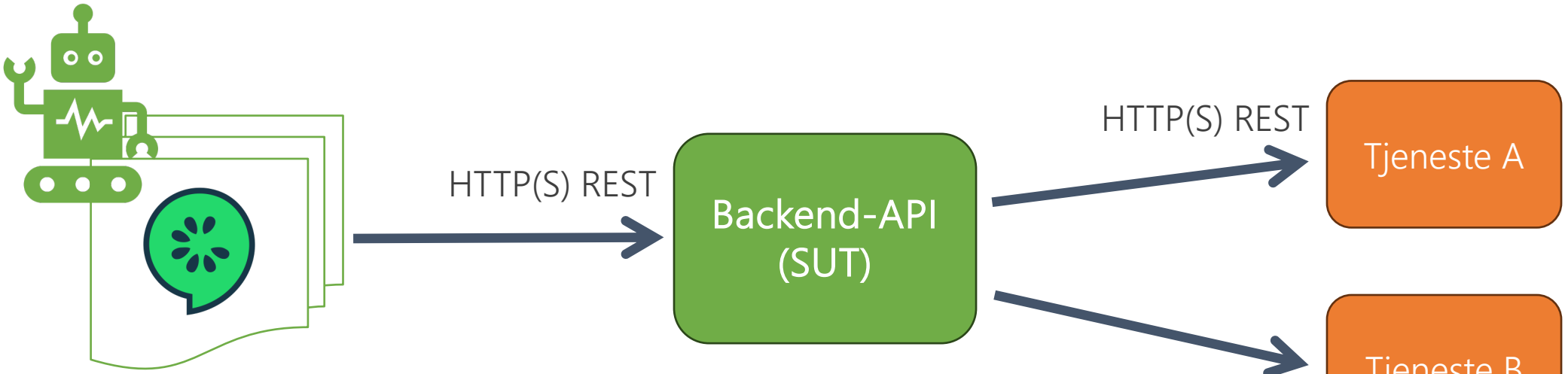
Generell problemstilling



Cucumber

- Verktøy for automatisert test av akseptansekriterier spesifisert i Gherkin-format
  - Format for tekstlig beskrivelse av forretningsregler
  - Gir et felles språk for utviklere, testere og fagpersoner
- Leser *.feature-filer* (ren tekst) og kobler hvert *steg* i scenariene til en *stegdefinisjon* i koden
- Stegdefinisjon = *Funksjon* i et programmeringsspråk + et *regulært uttrykk* (regex)
- Selve testkoden (stegdefinisjonene) kan implementeres på ***flere ulike testnivåer*** og ***med de biblioteker/rammeverk man selv ønsker***
- Implementasjon tilgjengelig for mange ulike programmeringsspråk

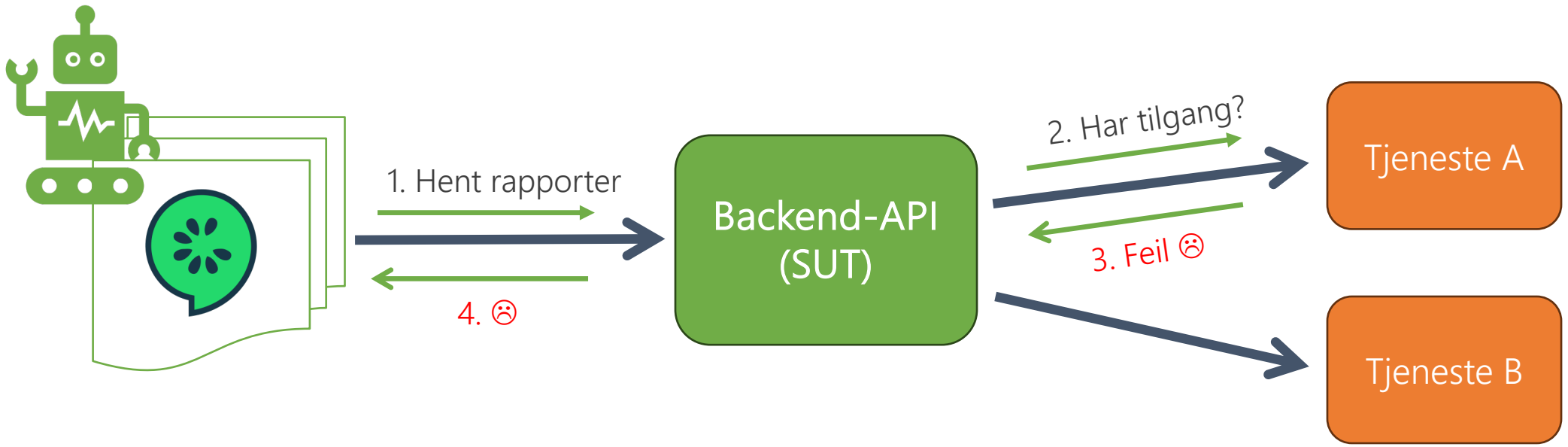
Eksempel på problemstilling



Scenario:

```
Gitt ...
Når ...
Så ...
```

Eksempel på problemstilling

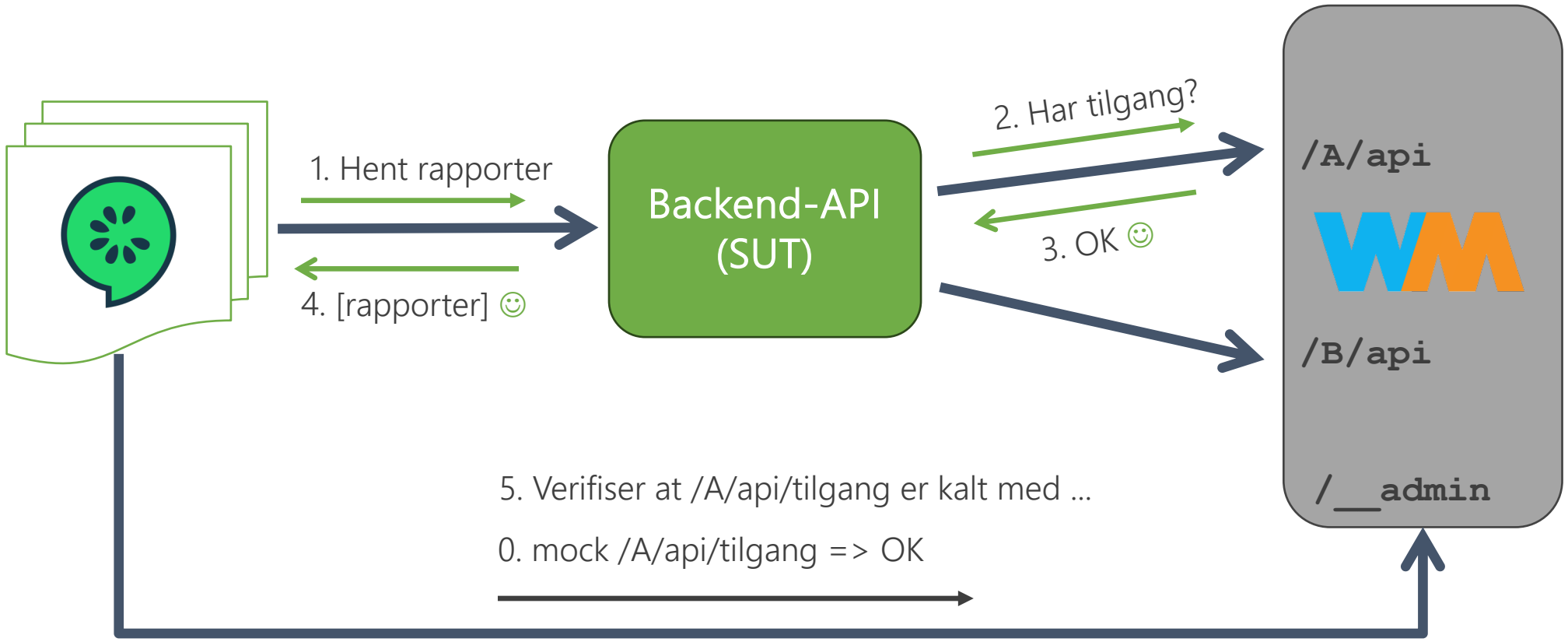



```
Scenario:
  Gitt at tjeneste A svarer med «Tilgang OK»
  Når jeg kaller SUT med «Hent rapporter»
  Så skal SUT sjekke min tilgang mot tjeneste A
  Og så skal SUT svare meg med en liste av rapporter
```

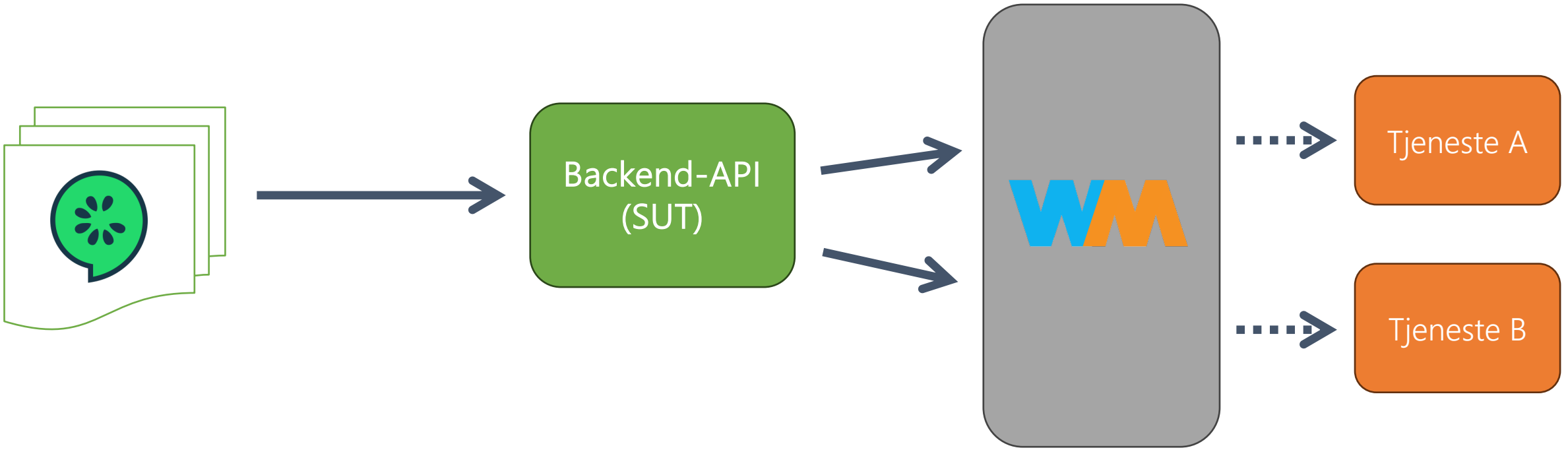
WireMock

- Løsning for mocking av REST-API for testing
  - Både stand-alone server og skytjeneste
  - API/bibliotek
- Stubbing/mocking av API-responser, basert på URL, header og body
  - Verifikasjon av mottatte kall
  - Record/playback
  - Konfigurerbare forsinkelser og feilresponser
  - Kan fungere som proxy
  - Støtte for tilstandsbasert oppførsel

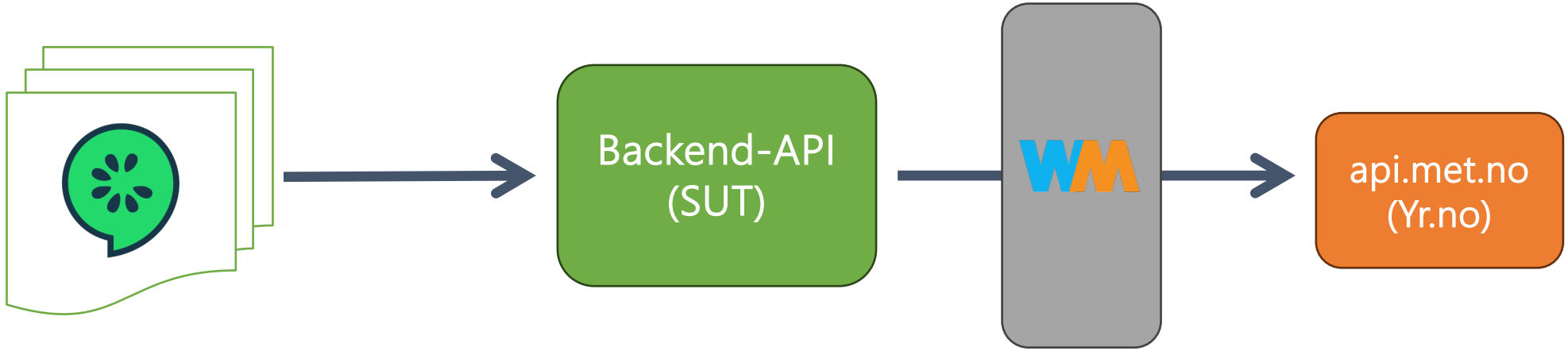
Løsning med WireMock



WireMock med proxying



Eksempelapplikasjon/-oppsett



- Enkel, hjemmesnekret applikasjonsserver med HTTP REST-grensesnitt
  - TypeScript, NodeJS, Express
- Overordnet funksjonalitet:
  - Hente værmelding fra Yr (met.no) for en navngitt lokasjon
  - Legge til og fjerne lokasjoner
  - Ingen persistering/permanent lagring

API for eksempelapplikasjon

- `GET /health` – Returner status OK så lenge applikasjonen er oppe
- `GET /health/yr` – Hent status fra yr.no og returner OK hvis svar
- `GET /locations` – List ut registrerte lokasjoner
- `POST /locations` – Legg til en ny lokasjon
  - Format: `{ name: string, coordinates: { lat: float, lon: float } }`
- `GET /locations/:name` – Hent en registrert lokasjon
- `DELETE /locations/:name` – Slett en registrert lokasjon
- `GET /weather/:locationName` – Hent en enkel værmelding for en registrert lokasjon

Eksempelkode

https://github.com/magnuseh/odin2023

- **/sut** – System under test (TypeScript, NodeJs, Express
- **/cucumber** – Black-box systemtester (TypeScript, NodeJs, CucumberJS)

Mer info i README.md

Oppgaver

1. Stegdefinisjon for helsesjekk

Scenarioet "Applikasjonen skal gi status ikke OK hvis Yr ikke svarer OK på status" i health.feature har en stegdefinisjon som ikke er implementert.

Finn og implementer dette steget.

2. Scenario og stegdefinisjon for feil ved henting av værmelding

Det finnes allerede et par scenarier for henting av værmelding for en gitt lokasjon, men dekningen er ikke god og bør utvides.

Deloppgaver:

1. Formulér et scenario der henting av værmelding fra Yr feiler
2. Implementer eventuelle manglende stegdefinisjoner

3. Scenarioer og stegdefinisjoner for ulike værtyper tjenesten kan returnere

SUT returnerer tre ulike forenklede værmeldinger basert på værdataene som returneres fra met.no/yr.no.

Deloppgaver:

1. Formulér scenarier som dekker denne funksjonaliteten
2. Implementer stegdefinisjoner med eventuelle mockdata nødvendig for automatisert, black-box systemtesting av dette

Tips: Se ./sut/src/models/weather.ts for definisjoner på de tre ulike værtypene

testify.no