MUSZĘ MIEĆ TO NA UMOWIE, INACZEJ NIE ROBIĘ

(PACT LVL 1)

Marcin Baranowski
Snowflake

Obraz autorstwa gpointstudiona Freepik

Dlaczego tak traktujemy nasz software?

Jak sprawić by testy były…

Jak sprawić by testy były…

stabilniejsze

Jak sprawić by testy były…

niezawodne

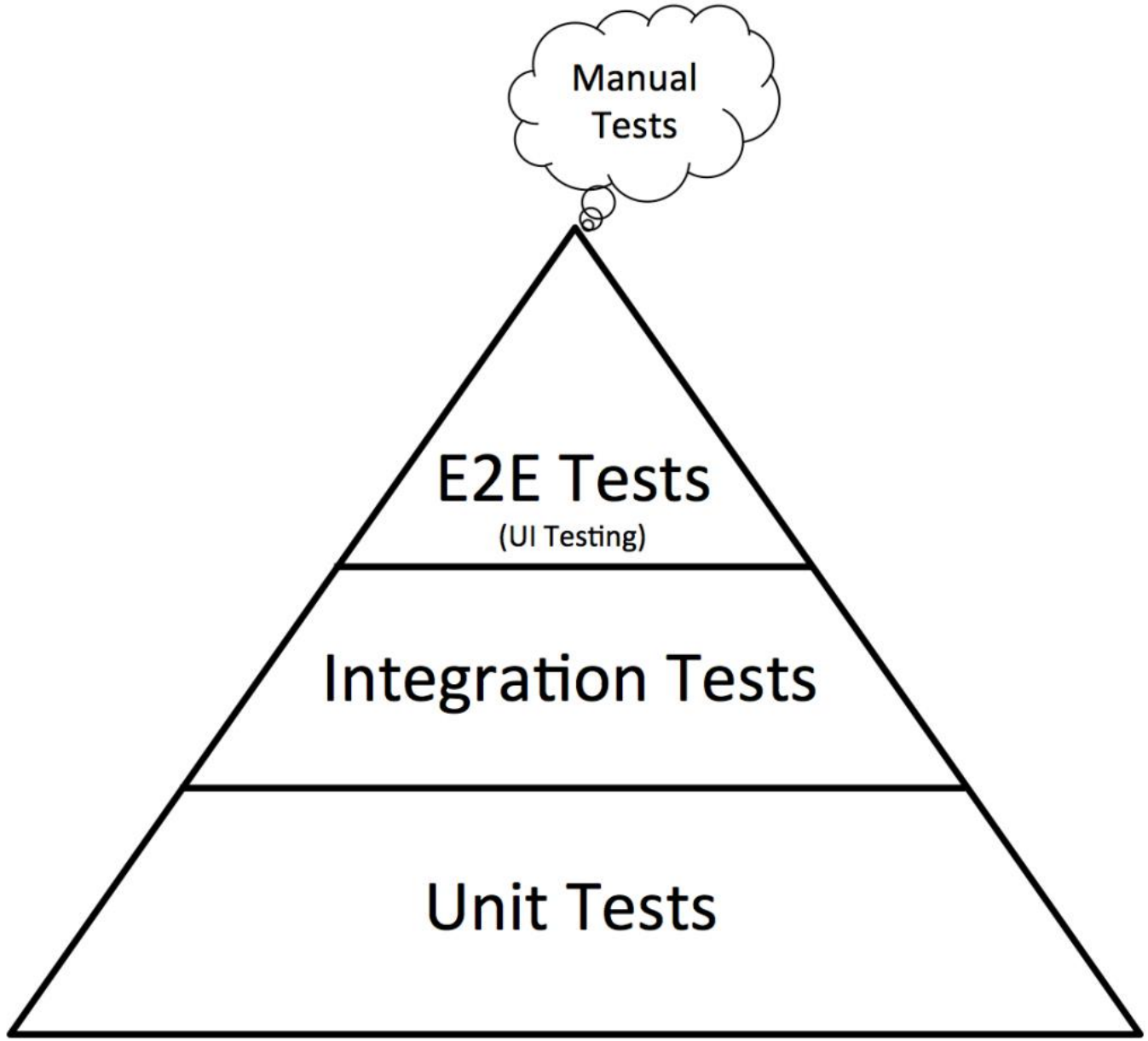
Jak sprawić by testy były…

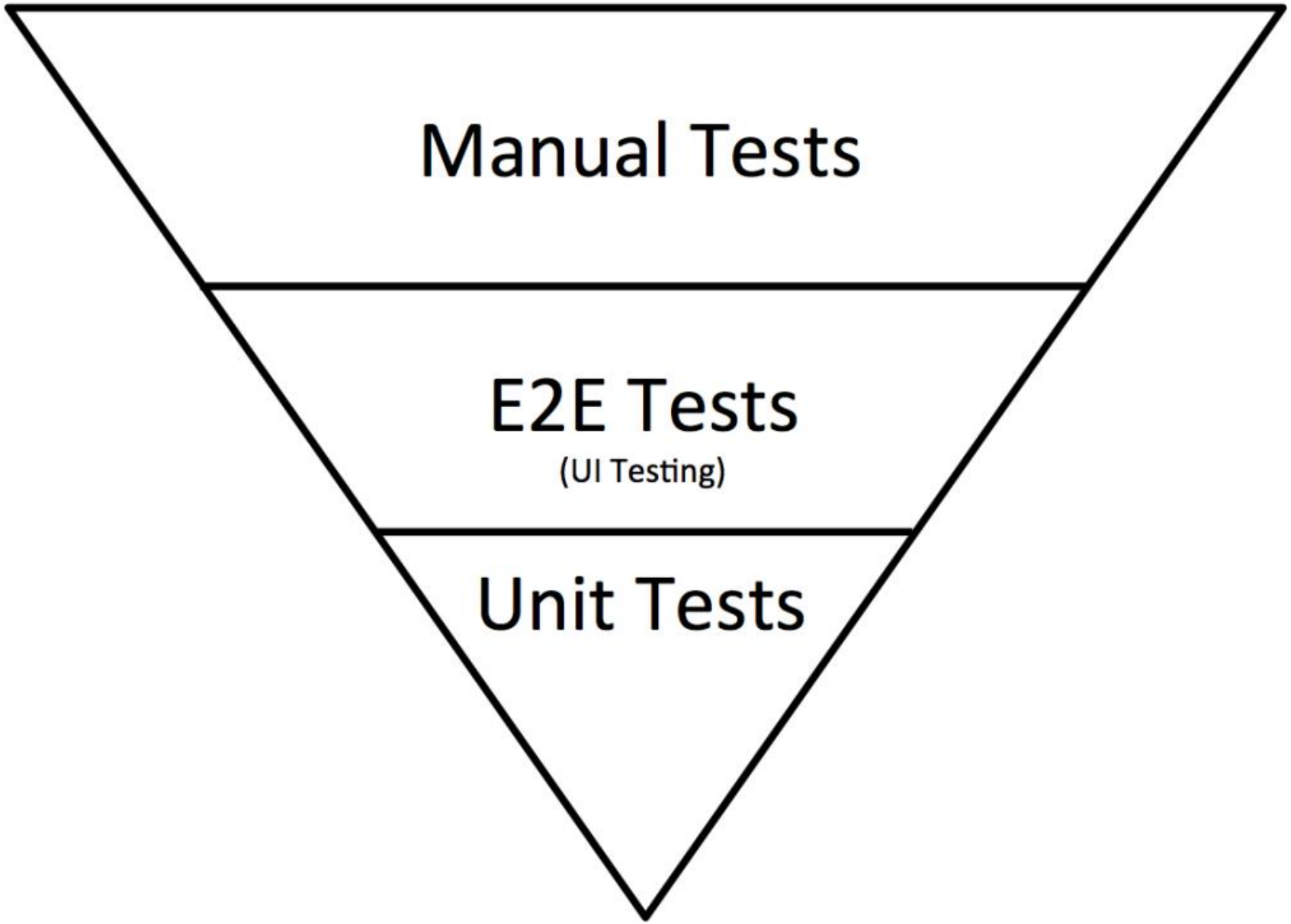
łatwiejsze do napisania

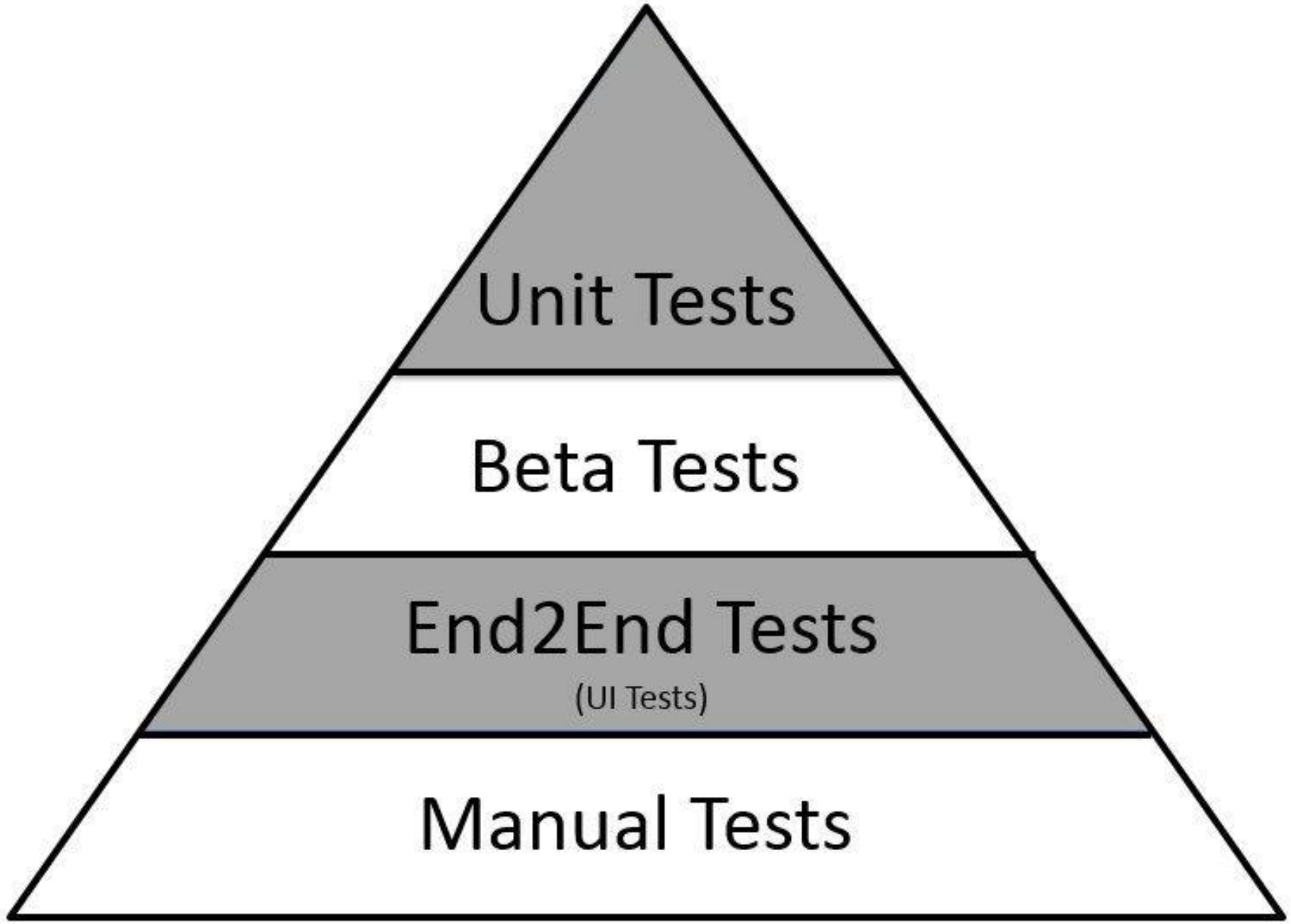
Jak sprawić by testy były…

szybsze

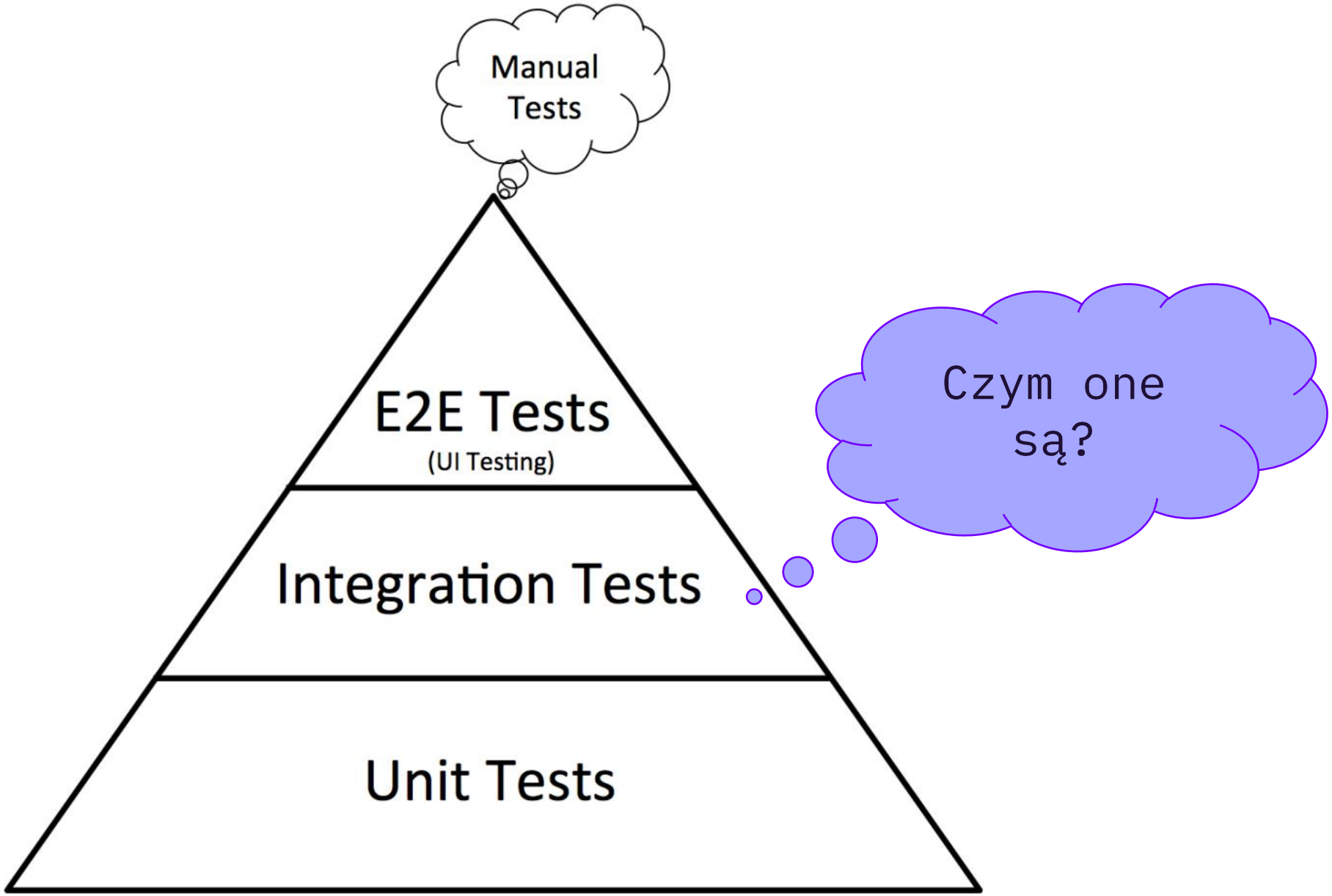
…a wdrożenia były tak łatwe…


Manual
Tests
E2E Tests
(UI Testing)
Integration Tests
Unit Tests


Manual Tests
E2E Tests
(UI Testing)
Unit Tests


Unit Tests
Beta Tests
End2End Tests
(UI Tests)
Manual Tests

Mam nadzieję
że po tej prezentacji
wszystko stanie się dla Was
bardziej skomplikowane


Manual Tests
E2E Tests
(UI Testing)
Integration Tests
Unit Tests
Czym one są?

Test integracyjny

Wiele składników

Trudny do zrobienia

Długi czas wykonania

Wyniki mogą być różne

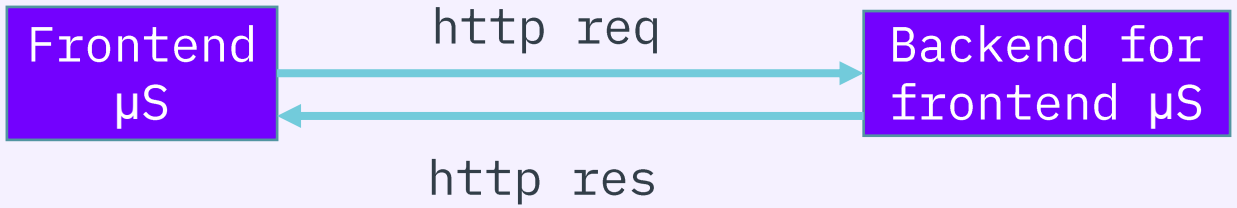
W kuchni może zostać bałagan

Testy integracyjne

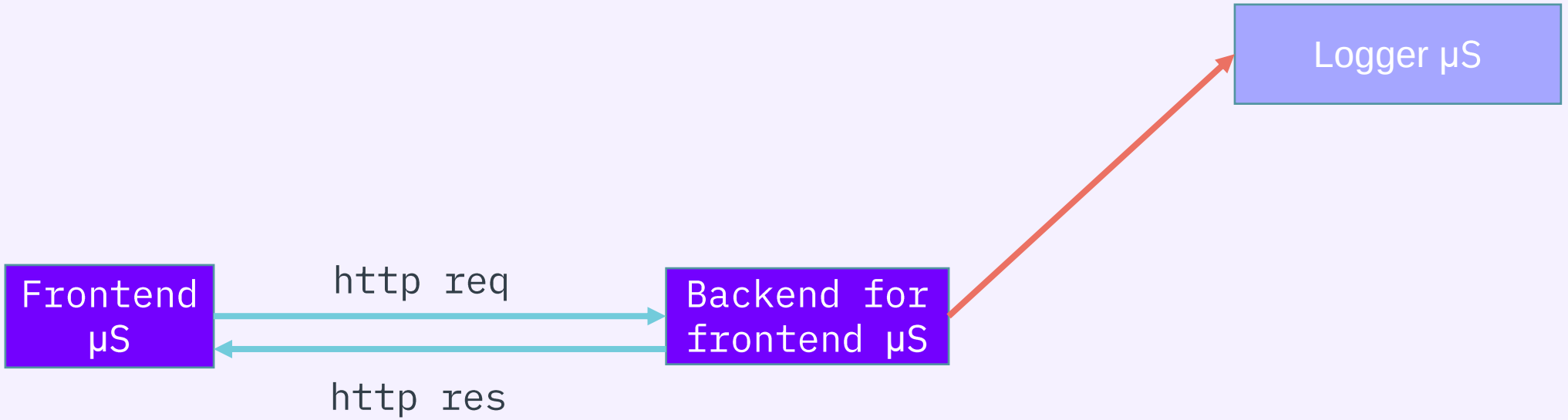
Testy integracyjne

Testy integracyjne

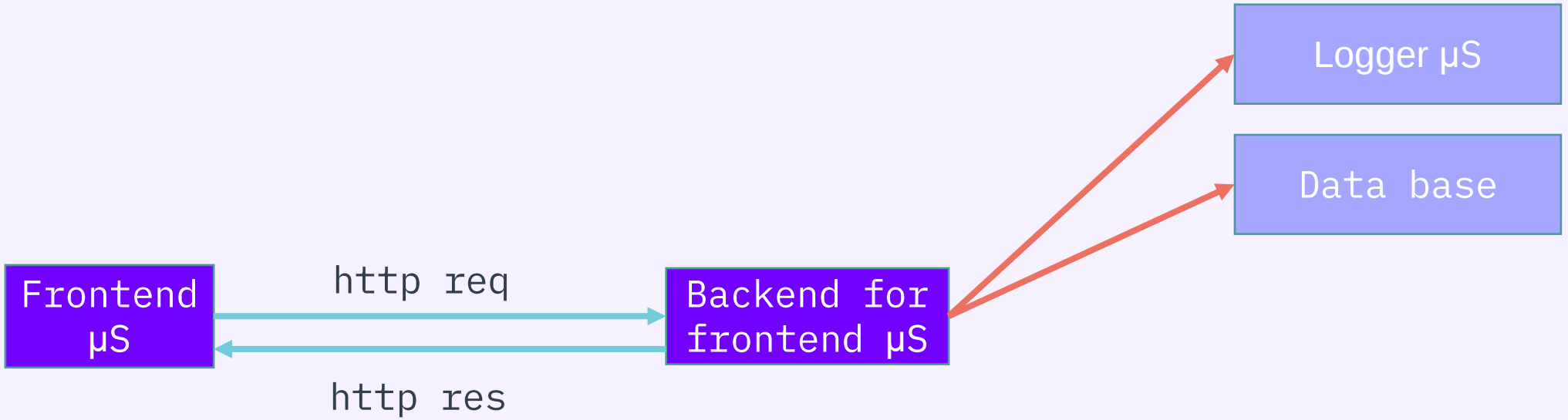
Testy integracyjne



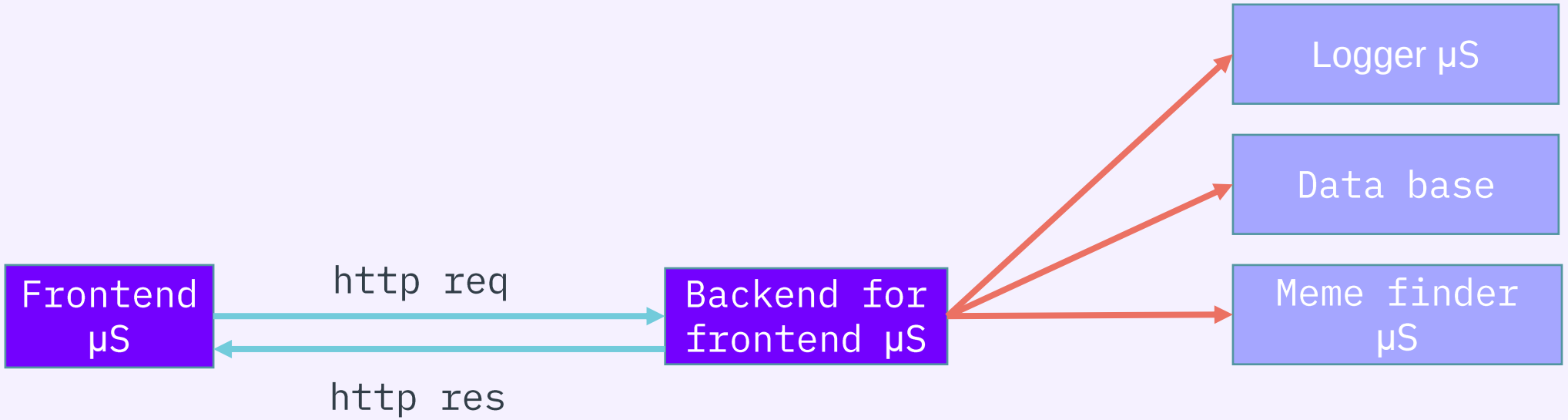
Testy integracyjne



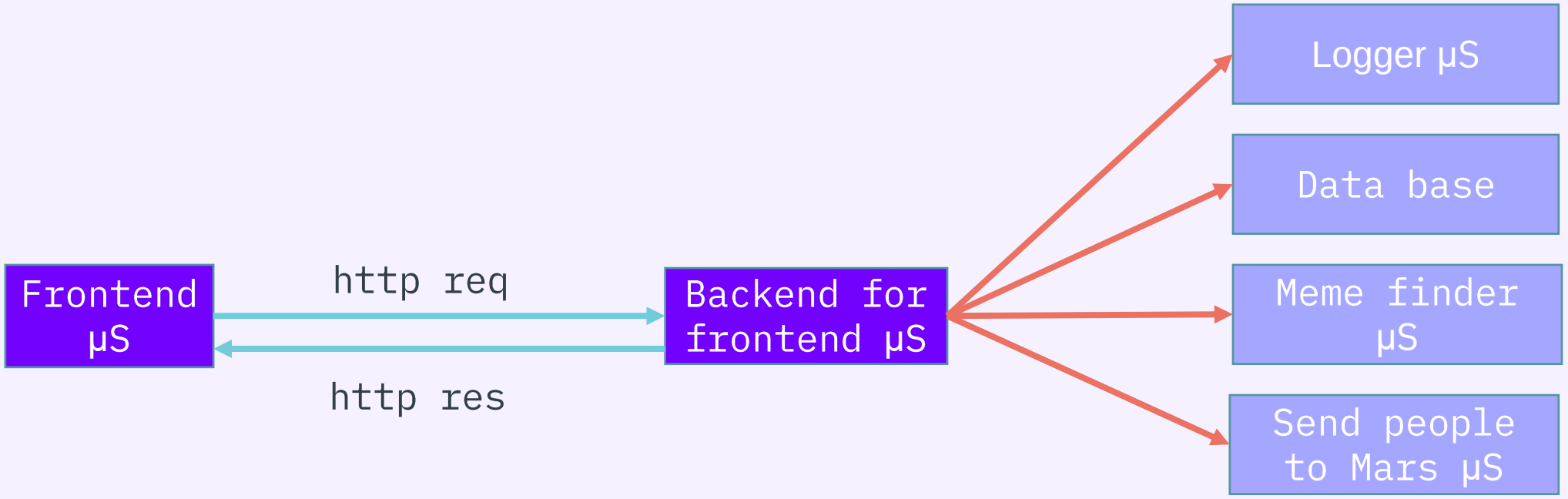
Testy integracyjne



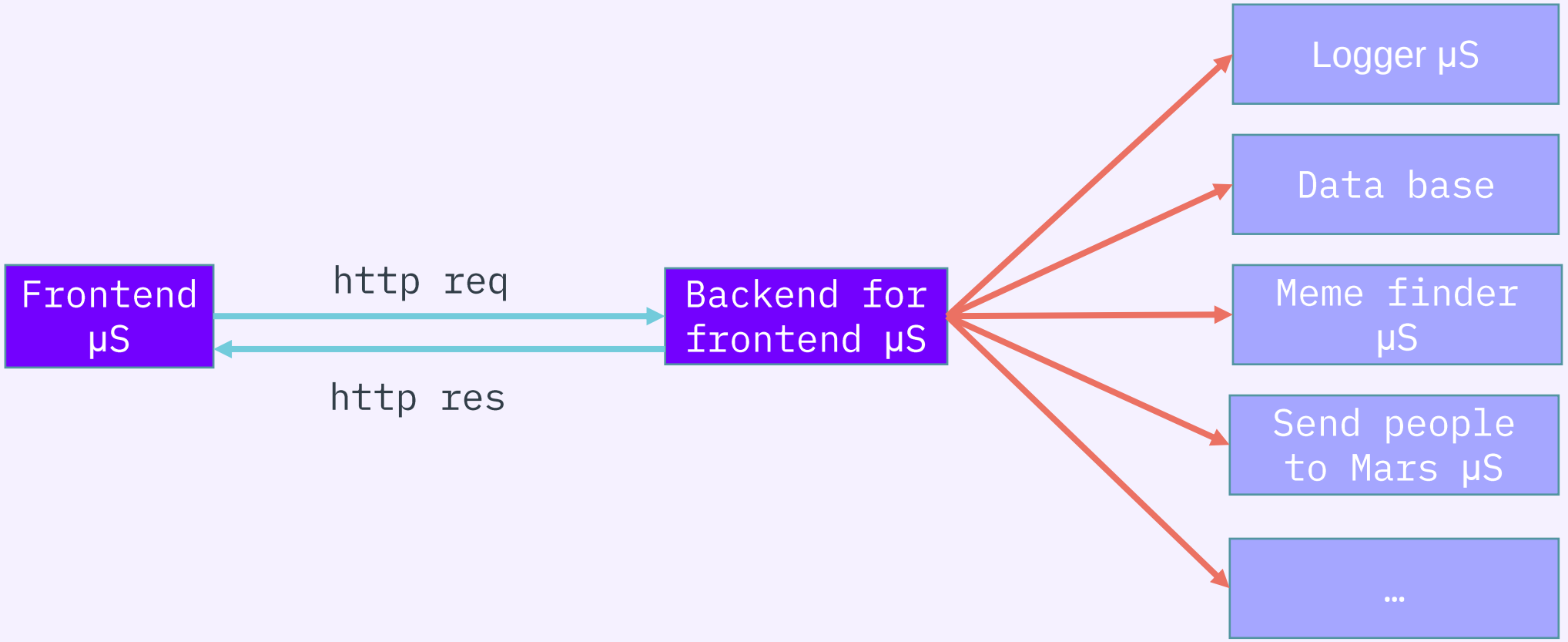
Testy integracyjne



Testy integracyjne

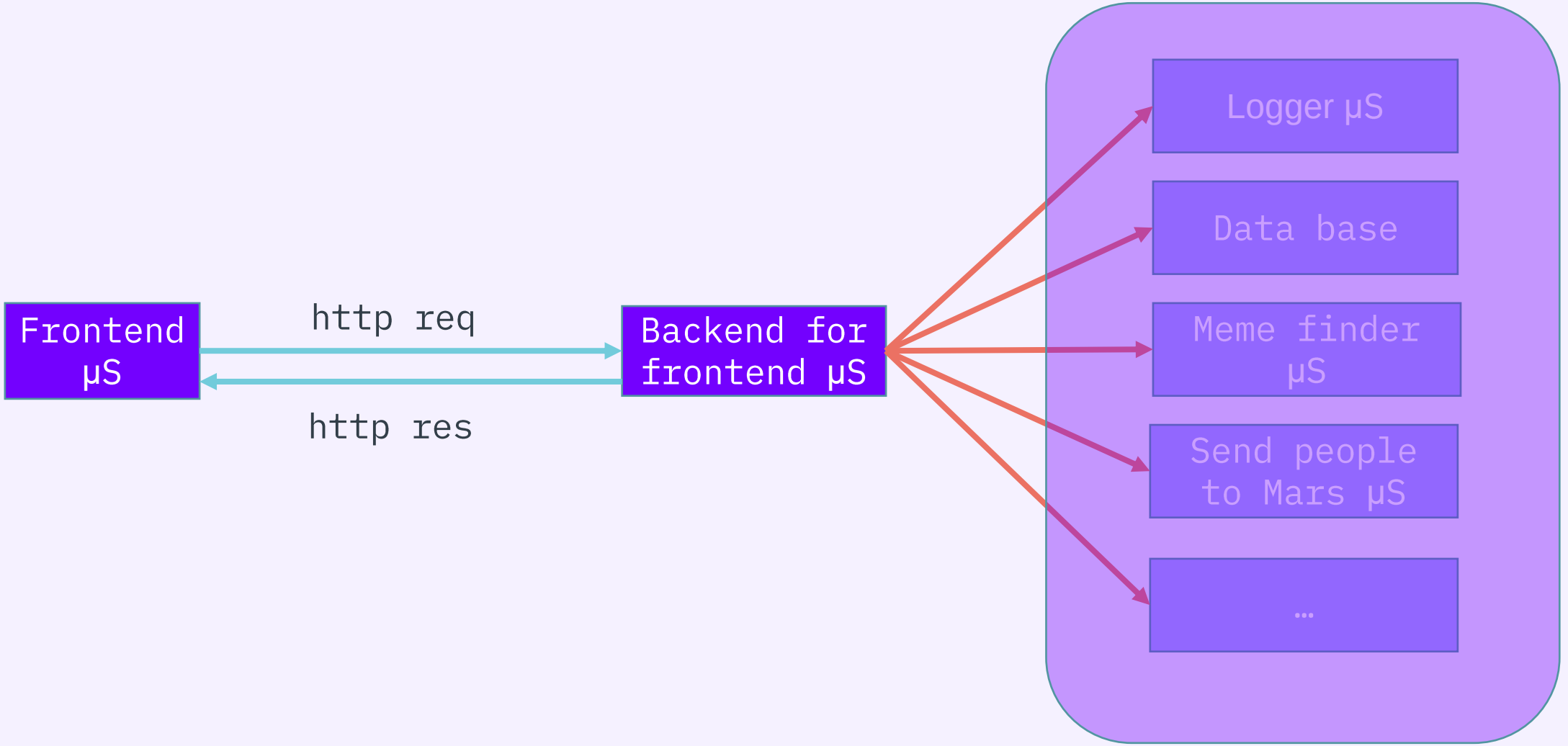


Testy integracyjne



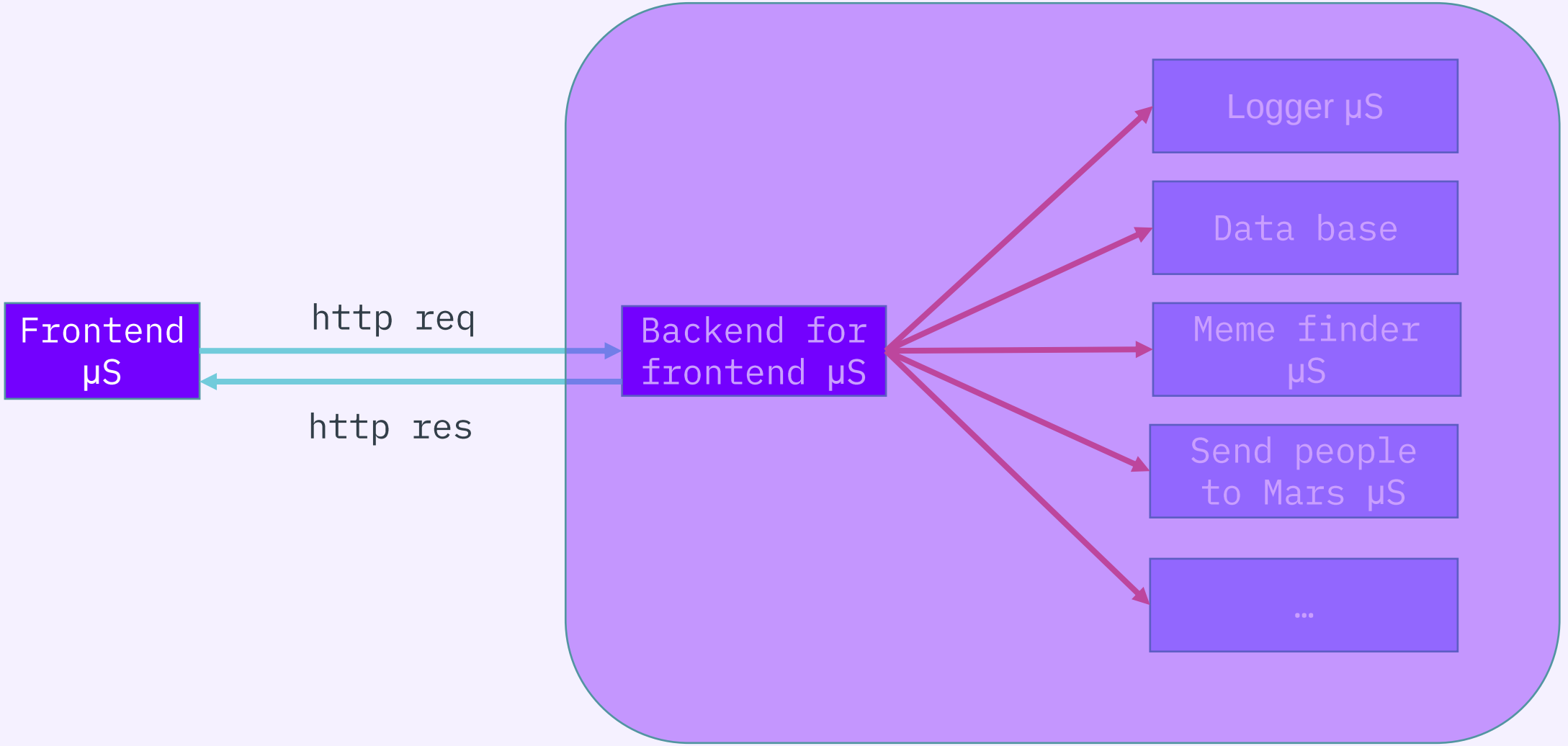
Testy integracyjne

Czy to jest potrzebne?



Testy integracyjne

Czy to jest potrzebne?



Czy mogę powierzyć Wam tajemnicę?

Bigos

Bigos nie

Bigos nie jest

Bigos nie jest jedynym

Bigos nie jest jedynym daniem

Bigos nie jest jedynym daniem na świecie!

Jest jeszcze…

Obraz autorstwa Racool_studio na Freepik

Testy kontraktowe

Tylko kilka składników

Łatwa do przygotowania

Szybki posiłek

Ciężko ją schrzanić

…CHYBA ŻE JESTEŚ…

coś chce

Konsument (consumer)

Dostawca (provider)

coś daje

Testy kontraktowe

Consumer

Testy kontraktowe

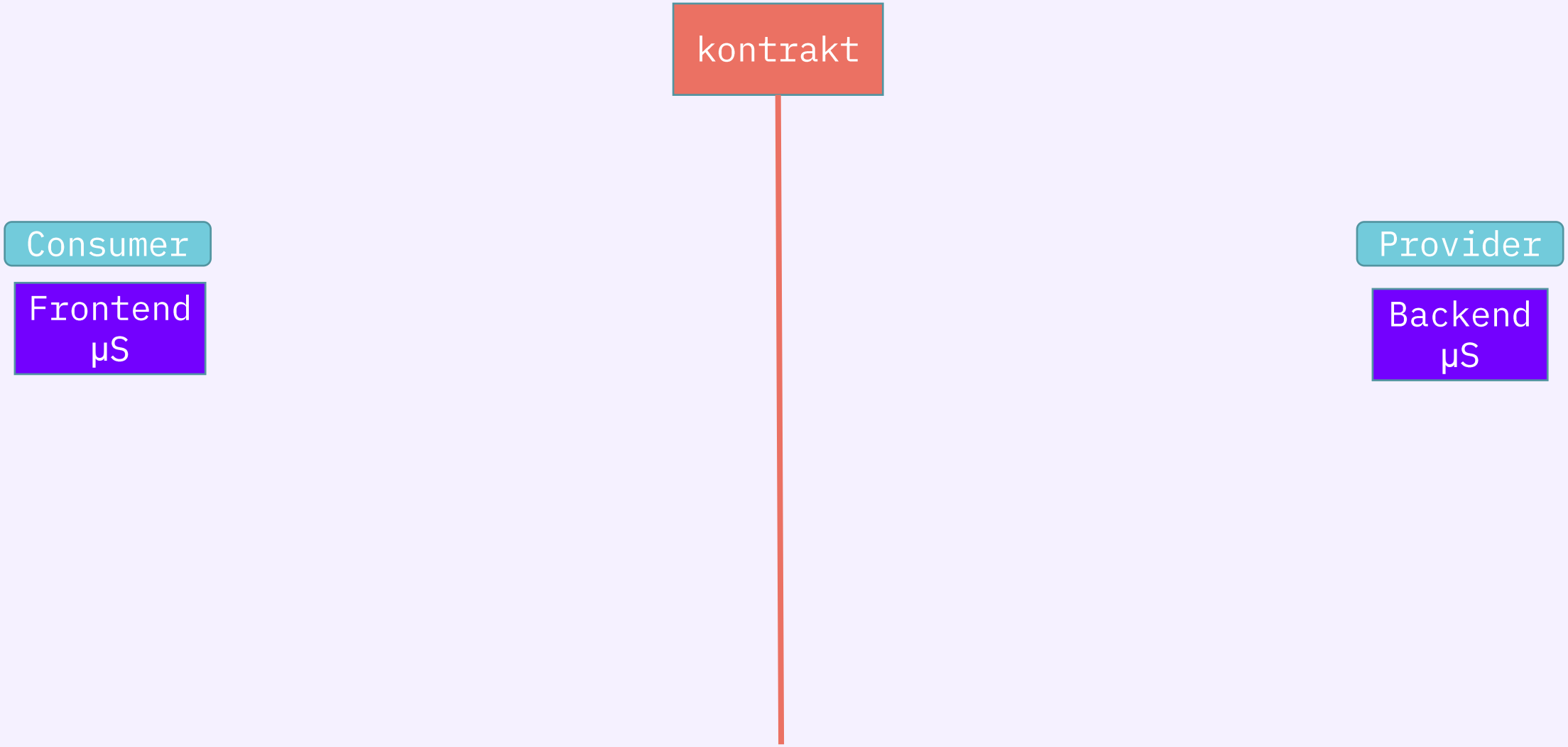
Consumer

Provider

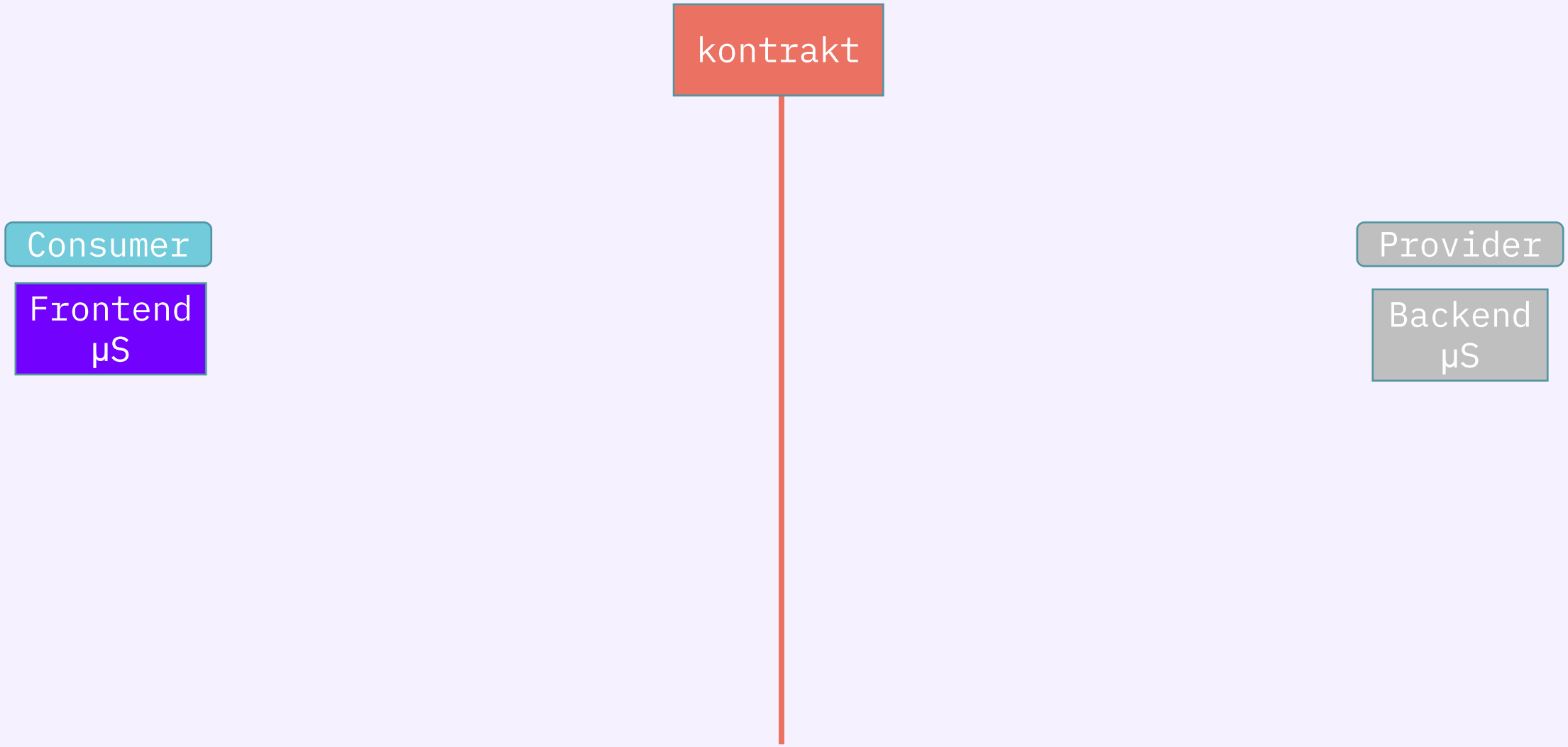
Testy kontraktowe

Testy kontraktowe

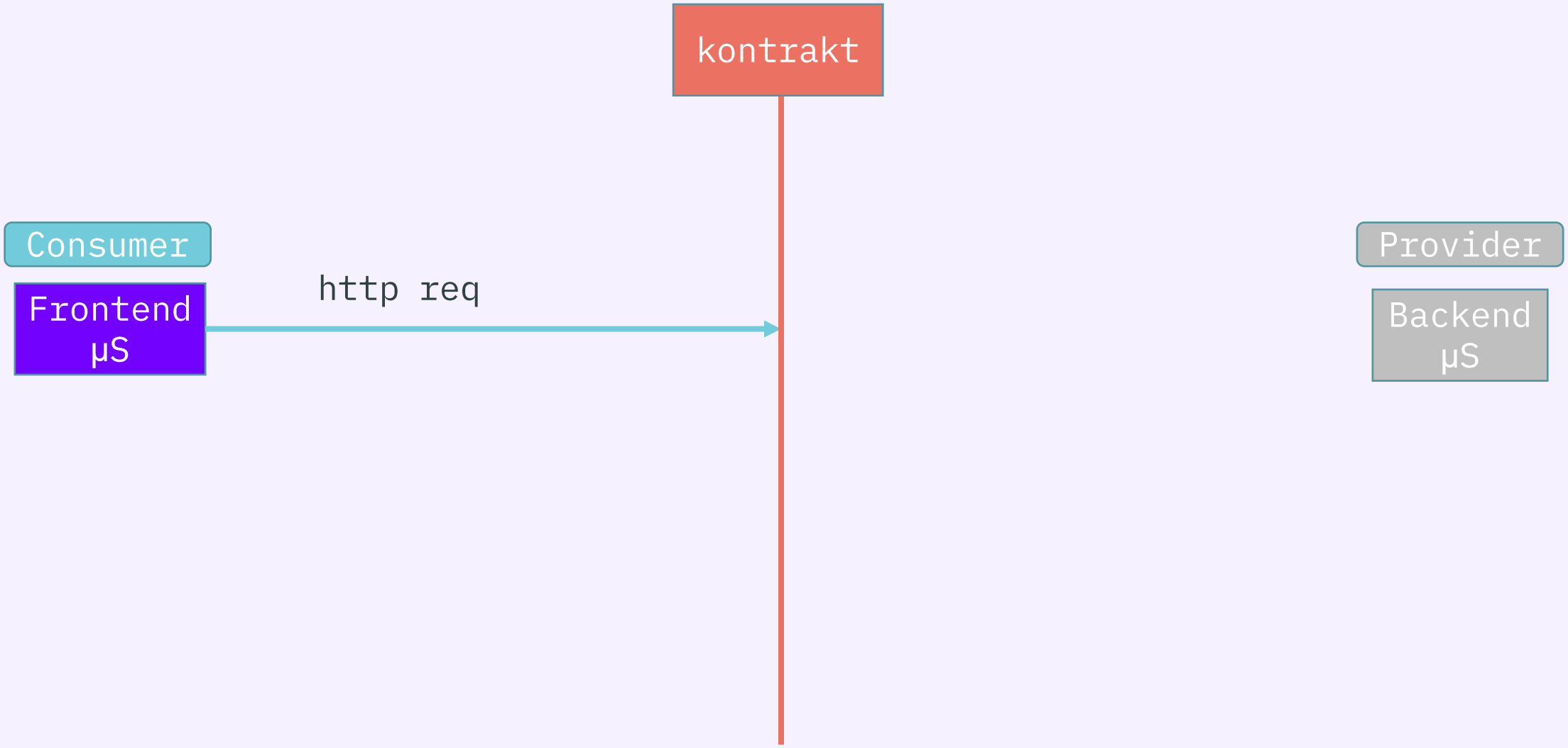
Testy kontraktowe



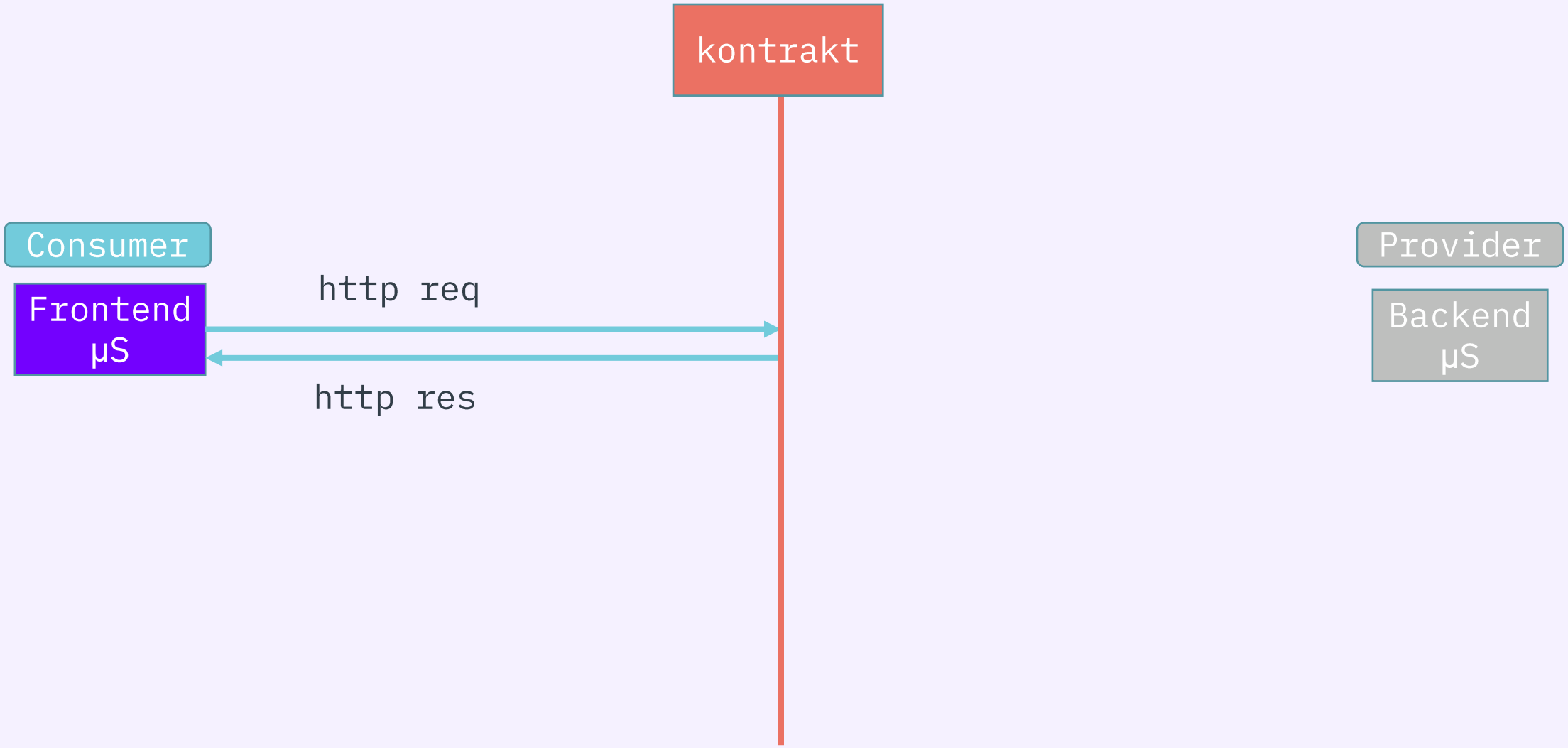
Testy kontraktowe – perspektywa konsumenta



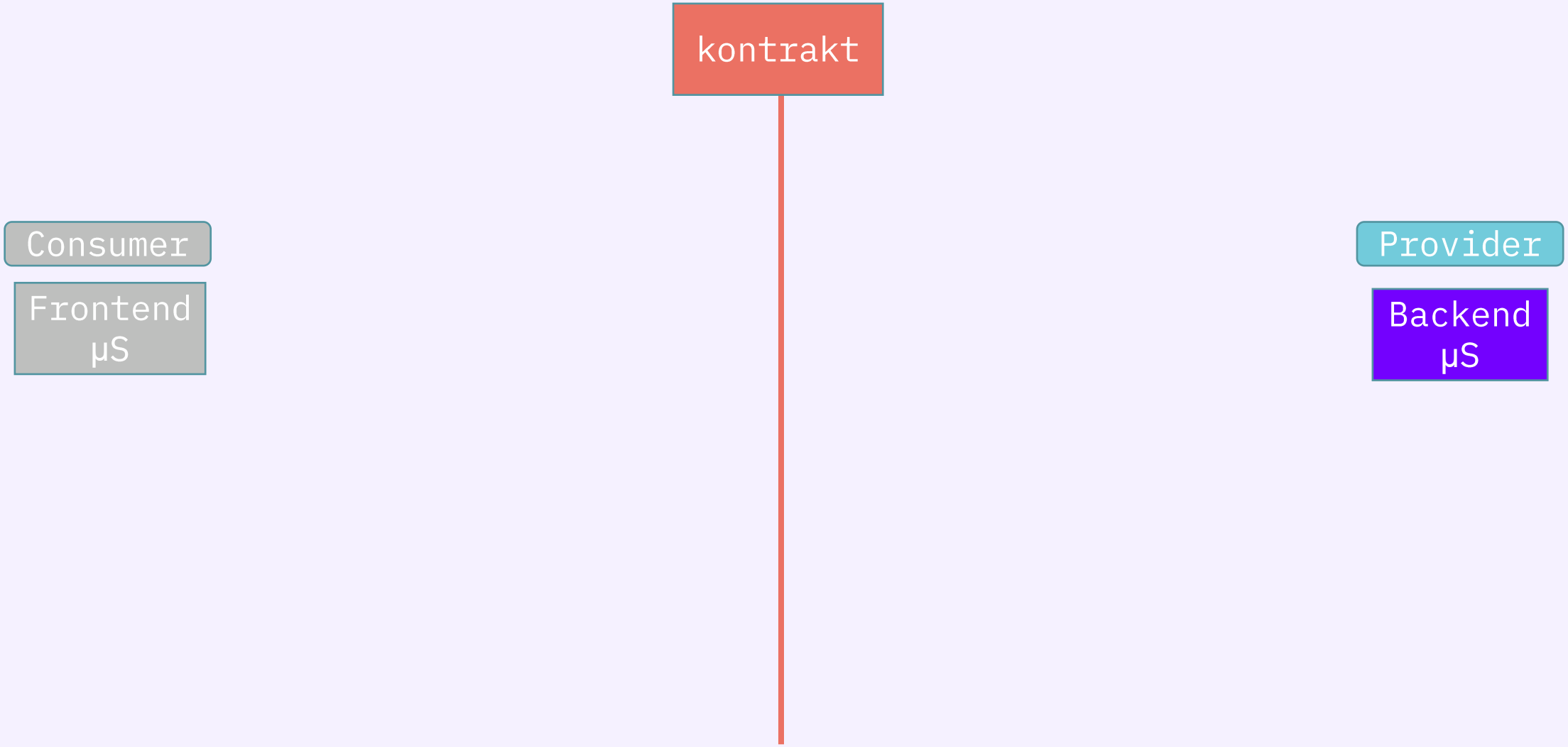
Testy kontraktowe – perspektywa konsumenta



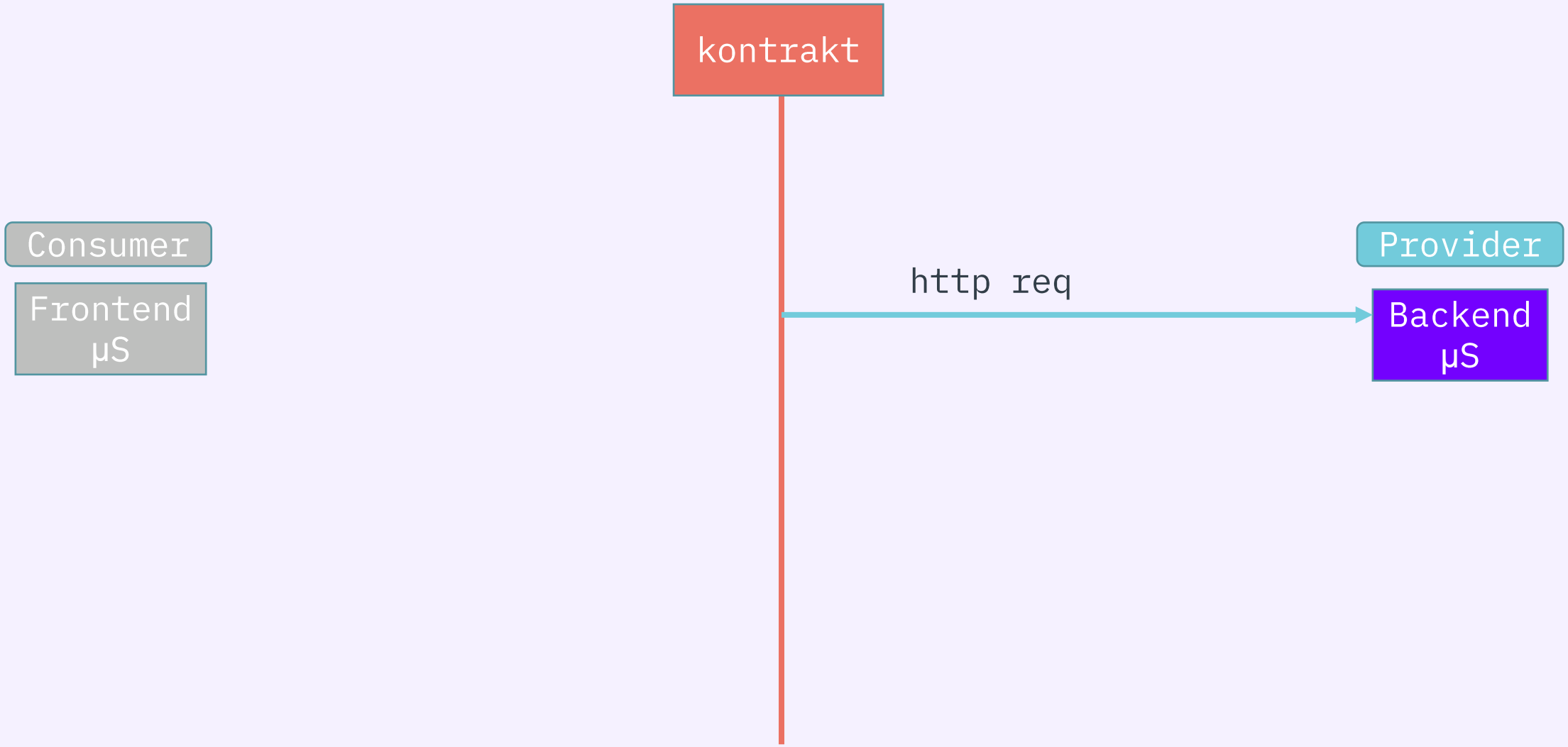
Testy kontraktowe – perspektywa konsumenta



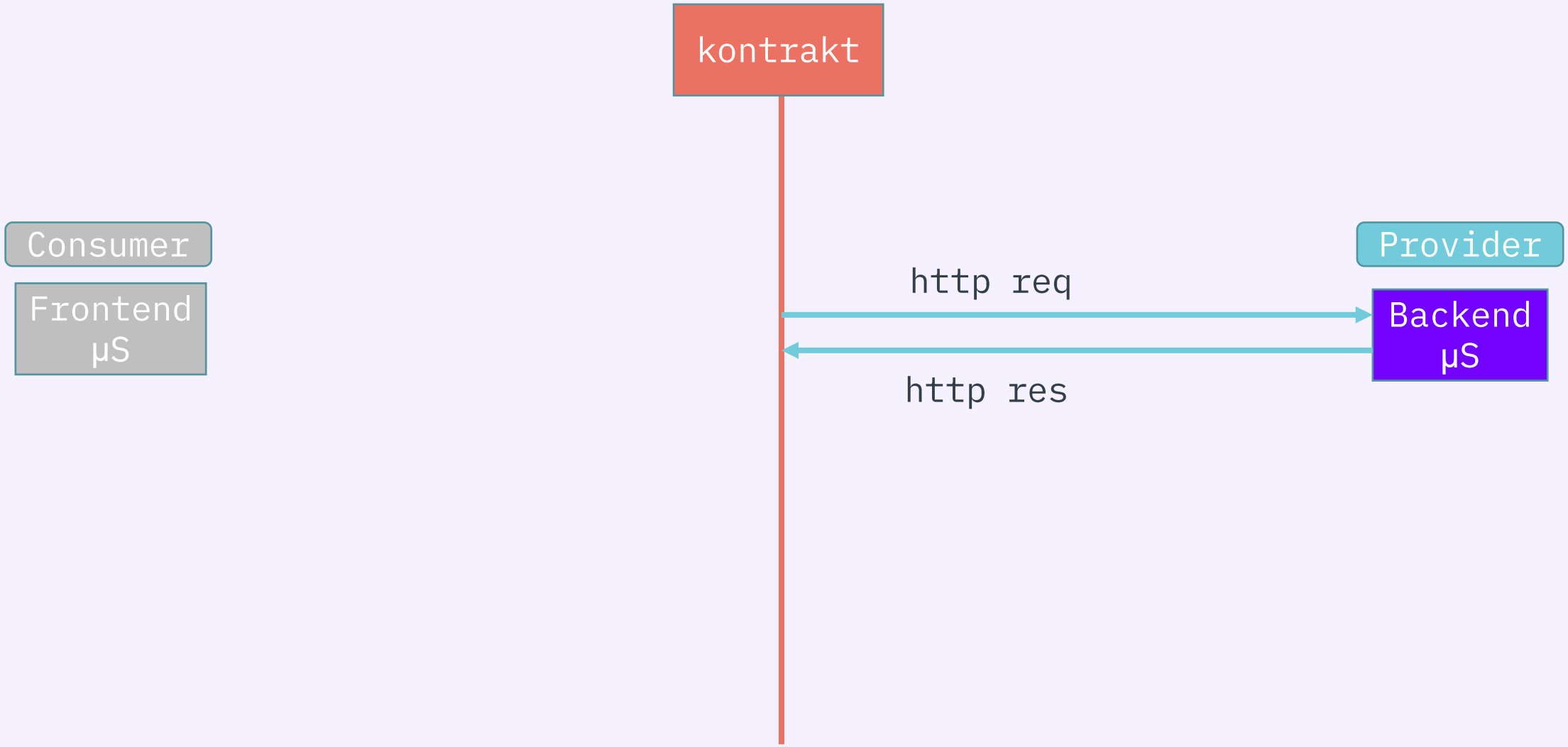
Testy kontraktowe – perspektywa producenta



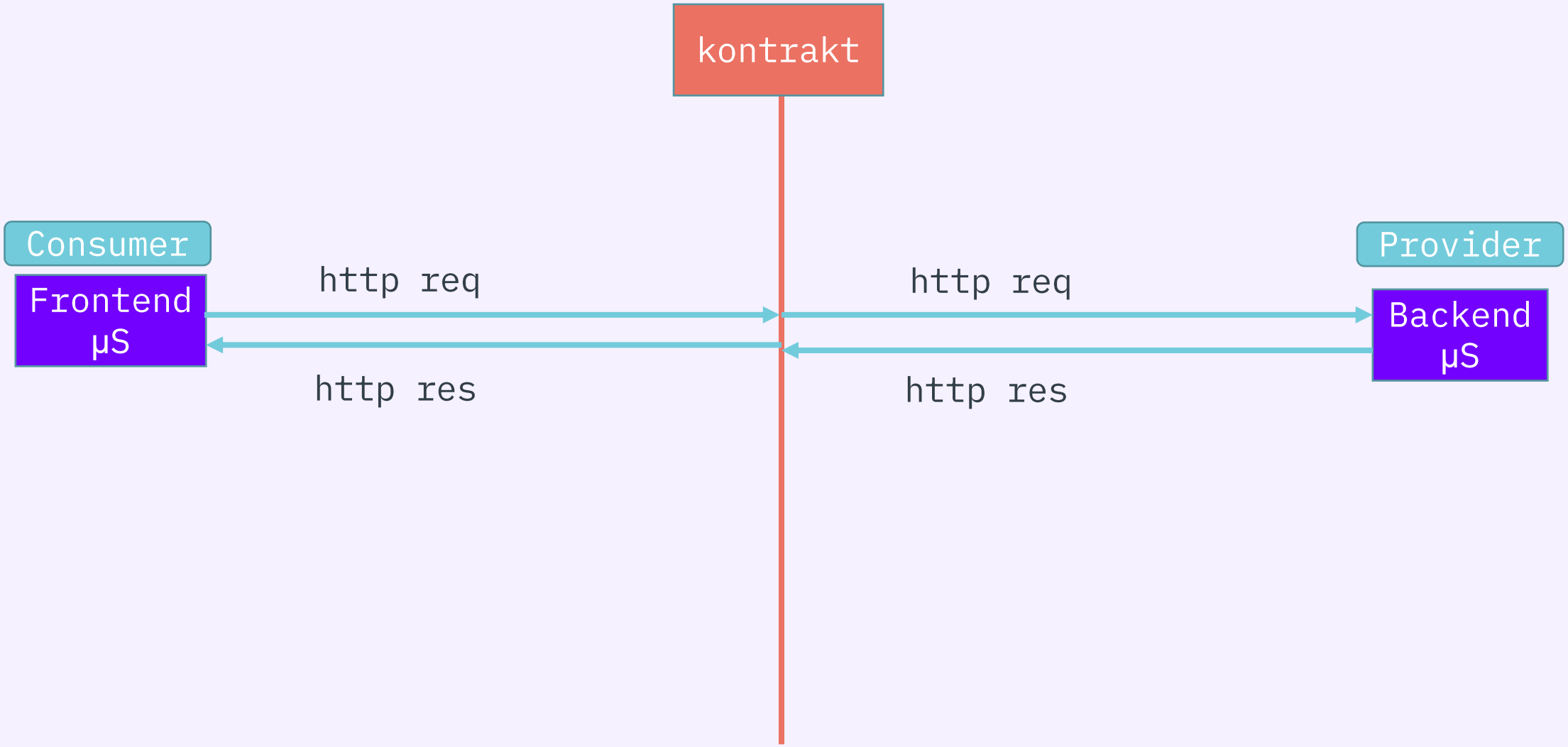
Testy kontraktowe – perspektywa producenta



Testy kontraktowe – perspektywa producenta



Testy kontraktowe



Czym jest kontrakt?

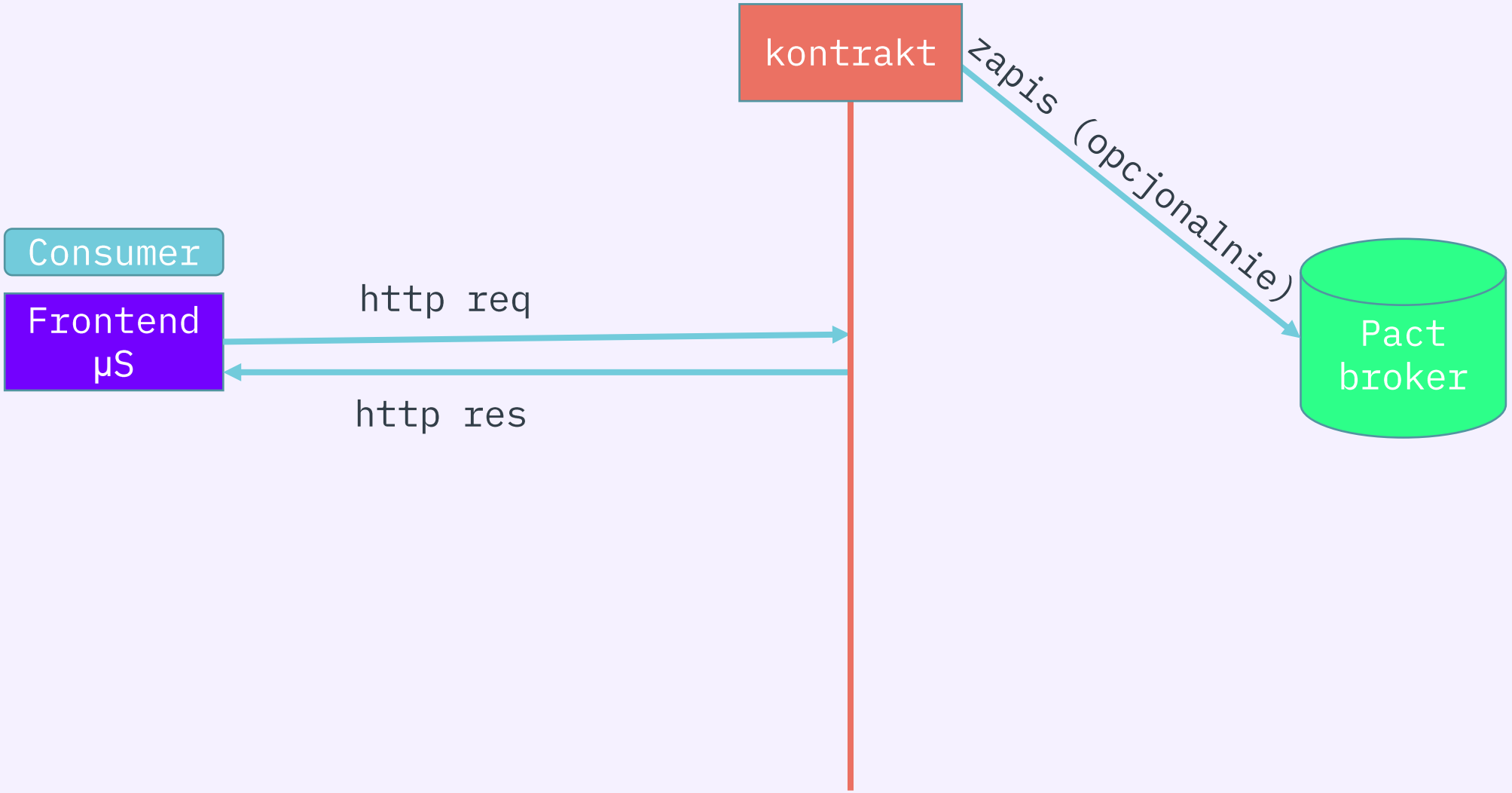
Po prostu json*

https://desciclopedia.org/wiki/Jason_Voorhees

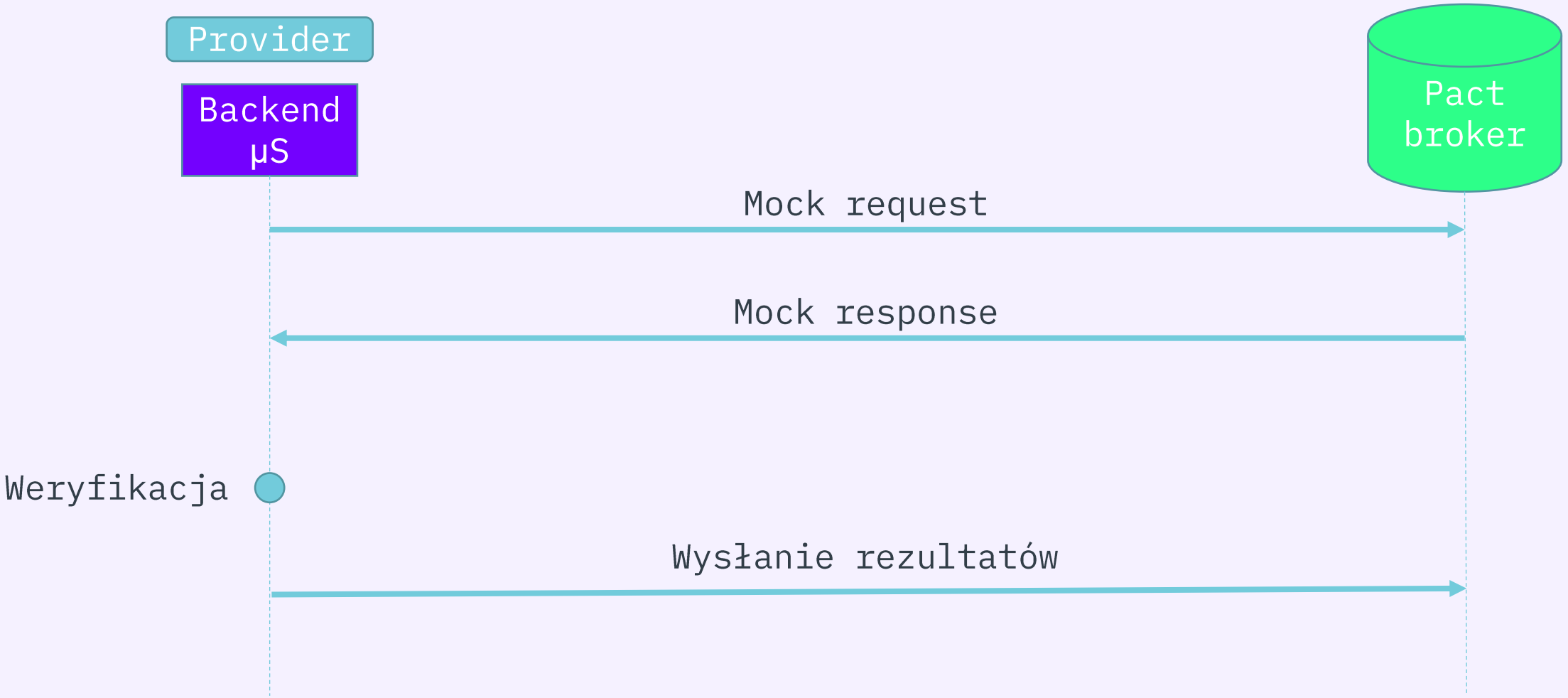
Jakie istnieją narzędzia do testów kontraktowych?

PACTS

Testy kontraktowe – Pact broker



Testy kontraktowe – Pact broker



LIVE DEMO

PRAWIE :P

Frontend

JavaScript/Axios/Mocha

Backend

Java/Spring/JUnit

Backend

```
GET  /dogs

GET  /dogs/{id}

POST /dogs
```

Frontend - request

```
exports.getMeDogs = endpoint => {

  return axios.request({



  })
}
```

Frontend - request

```
exports.getMeDogs = endpoint => {
  const url = endpoint.url
  const port = endpoint.port

  return axios.request({




  })
}
```

Frontend - request

```
exports.getMeDogs = endpoint => {
  const url = endpoint.url
  const port = endpoint.port

  return axios.request({
    method: "GET",
    baseURL: `${url}:${port}`,
    url: "/dogs",
    headers: { Accept: "application/json" },
  })
}
```

Frontend – setup testów

```
describe("Demo frontend Pact Test", () => {

  const provider = new Pact({

  })
```

Frontend - setup testów

```
describe("Demo frontend Pact Test", () => {

  const port = 8992


  const provider = new Pact({
    port: port,


  })
```

Frontend – setup testów

```
describe("Demo frontend Pact Test", () => {

  const port = 8992


  const provider = new Pact({
    port: port,
    log: path.resolve(process.cwd(), "logs", "mockserver-integration.log"),

  })
```

Frontend – setup testów

```
describe("Demo frontend Pact Test", () => {

  const port = 8992


  const provider = new Pact({
    port: port,
    log: path.resolve(process.cwd(), "logs", "mockserver-integration.log"),
    dir: path.resolve(process.cwd(), "pacts"),



  })
```

Frontend – setup testów

```
describe("Demo frontend Pact Test", () => {

  const port = 8992


  const provider = new Pact({
    port: port,
    log: path.resolve(process.cwd(), "logs", "mockserver-integration.log"),
    dir: path.resolve(process.cwd(), "pacts"),
    spec: 2,


  })
```

Frontend – setup testów

```
describe("Demo frontend Pact Test", () => {

  const port = 8992

const provider = new Pact({
    port: port,
    log: path.resolve(process.cwd(), "logs", "mockserver-integration.log"),
    dir: path.resolve(process.cwd(), "pacts"),
    spec: 2,
    consumer: "Demo frontend",
    provider: "Demo backend",
  })
```

Frontend – test kontraktowy

```
describe("get /dogs", () => {

  before(done => {
            const interaction = {


                withRequest: {

                },
                willRespondWith: {

                },
              }
  })
...
```

Frontend – test kontraktowy

```
describe("get /dogs", () => {



 before(done => {
            const interaction = {


              withRequest: {
                method: "GET",
                path: "/dogs",
              },
              willRespondWith: {

              },
            }
 })
...
```

Frontend – test kontraktowy

```
describe("get /dogs", () => {

  const EXPECTED_BODY = [...]

  before(done => {
                const interaction = {

                    withRequest: {
                      method: "GET",
                      path: "/dogs",
                    },
                    willRespondWith: {
                      status: 200,
                      headers: {
                        "Content-Type": "application/json",
                      },
                      body: EXPECTED_BODY,
                    },
                  }
  })
...
```

Frontend – test kontraktowy

```
describe("get /dogs", () => {

  const EXPECTED_BODY = [...]

  before(done => {
              const interaction = {

                  uponReceiving: "a request for all dogs",
                  withRequest: {
                    method: "GET",
                    path: "/dogs",
                  },
                  willRespondWith: {
                    status: 200,
                    headers: {
                      "Content-Type": "application/json",
                    },
                    body: EXPECTED_BODY,
                  },
                }
  })
...
```

Frontend – test kontraktowy

```
describe("get /dogs", () => {

  const EXPECTED_BODY = [...]

  before(done => {
              const interaction = {
                  state: "i have a list of dogs",
                  uponReceiving: "a request for all dogs",
                  withRequest: {
                    method: "GET",
                    path: "/dogs",
                  },
                  willRespondWith: {
                    status: 200,
                    headers: {
                      "Content-Type": "application/json",
                    },
                    body: EXPECTED_BODY,
                  },
                }
  })
...
```

Frontend – test kontraktowy

```
it("returns list of dogs", done => {
const urlAndPort = {
    url: url,
    port: port,
  }
  getMeDogs(urlAndPort).then(response => {
    expect(response.data).to.eql(EXPECTED_BODY)
    done()
  }, done)
})
```

Backend – klasa Dog

```
public class Dog {
  @Id
  @GeneratedValue
  @JsonIgnore
  private Long id;

  private String name;
  private int age;

  ...
```

Backend – kontroler

```
@RestController
public class DogsController {

  @Autowired
  private DogRepository dogRepository;

  @GetMapping("/dogs")
  public Iterable<Dog> allDogs() {

  }

 ...
}
```

Backend – kontroler

```
@RestController
public class DogsController {

  @Autowired
  private DogRepository dogRepository;

  @GetMapping("/dogs")
  public Iterable<Dog> allDogs() {

  }

  @PostMapping(value = "/dogs", consumes = "application/json")
  public ResponseEntity save(@RequestBody Dog dog) {


  }

  ...
}
```

Backend – kontroler

```
@RestController
public class DogsController {

  @Autowired
  private DogRepository dogRepository;

  @GetMapping("/dogs")
  public Iterable<Dog> allDogs() {

  }

  @PostMapping(value = "/dogs", consumes = "application/json")
  public ResponseEntity save(@RequestBody Dog dog) {


  }

  @GetMapping(value = "/dogs/{id}", produces = "application/json")
  public Dog dogById(@PathVariable("id") Long id) {

  }
}
```

Backend – kontroler

```
@RestController
public class DogsController {

  @Autowired
  private DogRepository dogRepository;

  @GetMapping("/dogs")
  public Iterable<Dog> allDogs() {
    return dogRepository.findAll();
  }

  @PostMapping(value = "/dogs", consumes = "application/json")
  public ResponseEntity save(@RequestBody Dog dog) {



  }

  @GetMapping(value = "/dogs/{id}", produces = "application/json")
  public Dog dogById(@PathVariable("id") Long id) {

  }
}
```

Backend – kontroler

```
@RestController
public class DogsController {

  @Autowired
  private DogRepository dogRepository;

  @GetMapping("/dogs")
  public Iterable<Dog> allDogs() {
    return dogRepository.findAll();
  }

  @PostMapping(value = "/dogs", consumes = "application/json")
  public ResponseEntity save(@RequestBody Dog dog) {
    Dog savedDog = dogRepository.save(dog);

  }

  @GetMapping(value = "/dogs/{id}", produces = "application/json")
  public Dog dogById(@PathVariable("id") Long id) {

  }
}
```

Backend – kontroler

```
@RestController
public class DogsController {

  @Autowired
  private DogRepository dogRepository;

  @GetMapping("/dogs")
  public Iterable<Dog> allDogs() {
    return dogRepository.findAll();
  }

  @PostMapping(value = "/dogs", consumes = "application/json")
  public ResponseEntity save(@RequestBody Dog dog) {
    Dog savedDog = dogRepository.save(dog);
    return ResponseEntity.created(URI.create("/dogs/" + savedDog.getId())).build();
  }

  @GetMapping(value = "/dogs/{id}", produces = "application/json")
  public Dog dogById(@PathVariable("id") Long id) {

  }
}
```

Backend – kontroler

```
@RestController
public class DogsController {

  @Autowired
  private DogRepository dogRepository;

  @GetMapping("/dogs")
  public Iterable<Dog> allDogs() {
    return dogRepository.findAll();
  }

  @PostMapping(value = "/dogs", consumes = "application/json")
  public ResponseEntity save(@RequestBody Dog dog) {
    Dog savedDog = dogRepository.save(dog);
    return ResponseEntity.created(URI.create("/dogs/" + savedDog.getId())).build();
  }

  @GetMapping(value = "/dogs/{id}", produces = "application/json")
  public Dog dogById(@PathVariable("id") Long id) {
    return dogRepository.findById(id).orElseThrow(DogNotFoundException::new);
  }
}
```

Backend – setup testów

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@Provider("Demo backend")
@PactBroker
public class PactVerificationTest {
...
```

Backend – setup testów

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@Provider("Demo backend")
@PactBroker
public class PactVerificationTest {
   @LocalServerPort
   private int port;

   @BeforeEach
   void setup(PactVerificationContext context) {
      context.setTarget(new HttpTestTarget("localhost", port));
   }

   @TestTemplate
   @ExtendWith(PactVerificationSpringProvider.class)
   void pactVerificationTestTemplate(PactVerificationContext context) {
      context.verifyInteraction();
   }
...
```

Backend – state

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@Provider("Demo backend")
@PactBroker
public class PactVerificationTest {
    ...

    @State(value = "i have a list of dogs")
    void dogsExists() {

    }

    ...
}
```

Backend – state

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@Provider("Demo backend")
@PactBroker
public class PactVerificationTest {
   @MockBean
   private DogRepository dogRepository;

   @State(value = "i have a list of dogs")
   void dogsExists() {
      when(dogRepository.findAll())


   }

   ...
}
```

Backend – state

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@Provider("Demo backend")
@PactBroker
public class PactVerificationTest {
   @MockBean
   private DogRepository dogRepository;

   @State(value = "i have a list of dogs")
   void dogsExists() {
      when(dogRepository.findAll()).thenReturn(Arrays.asList(

      ));
   }

   ...
}
```

Backend – state

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@Provider("Demo backend")
@PactBroker
public class PactVerificationTest {
   @MockBean
   private DogRepository dogRepository;

   @State(value = "i have a list of dogs")
   void dogsExists() {
      when(dogRepository.findAll()).thenReturn(Arrays.asList(
            new Dog(1L, "Max", 3,
            new Dog(200L, "Lassie", 8)
      ));
   }

   ...
}
```

Backend – state

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@Provider("Demo backend")
@PactBroker
public class PactVerificationTest {
   @MockBean
   private DogRepository dogRepository;

   @State(value = "i have a list of dogs")
   void dogsExists() {
      Dog max = new Dog(1L, "Max", 3);
      when(dogRepository.findAll()).thenReturn(Arrays.asList(
            max,
            new Dog(200L, "Lassie", 8)
      ));
      when(dogRepository.findById(anyLong())).thenReturn(Optional.of(max));
      when(dogRepository.save(any())).thenReturn(Optional.of(max));
   }
   ...
}
```

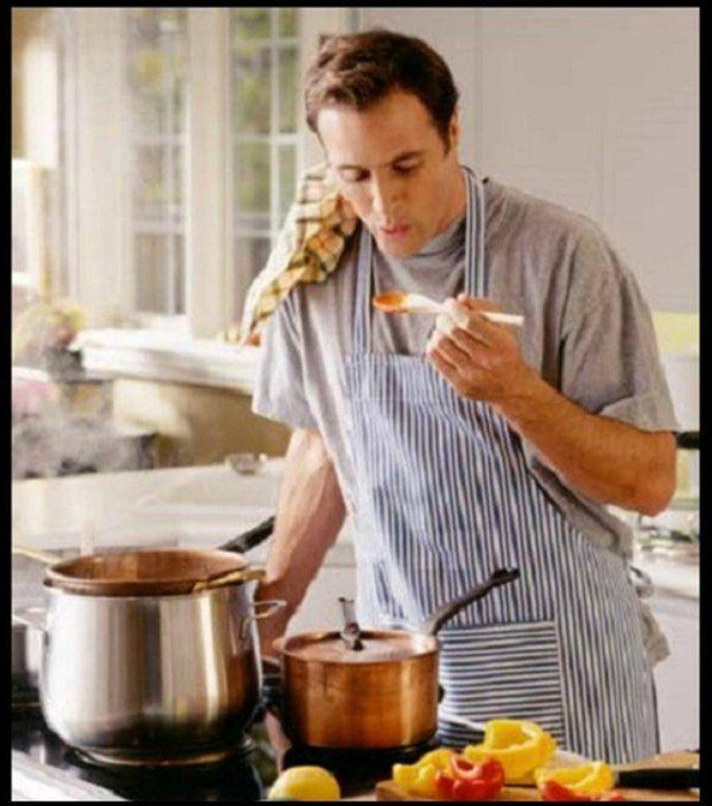
LIVE DEMO

(teraz naprawdę)

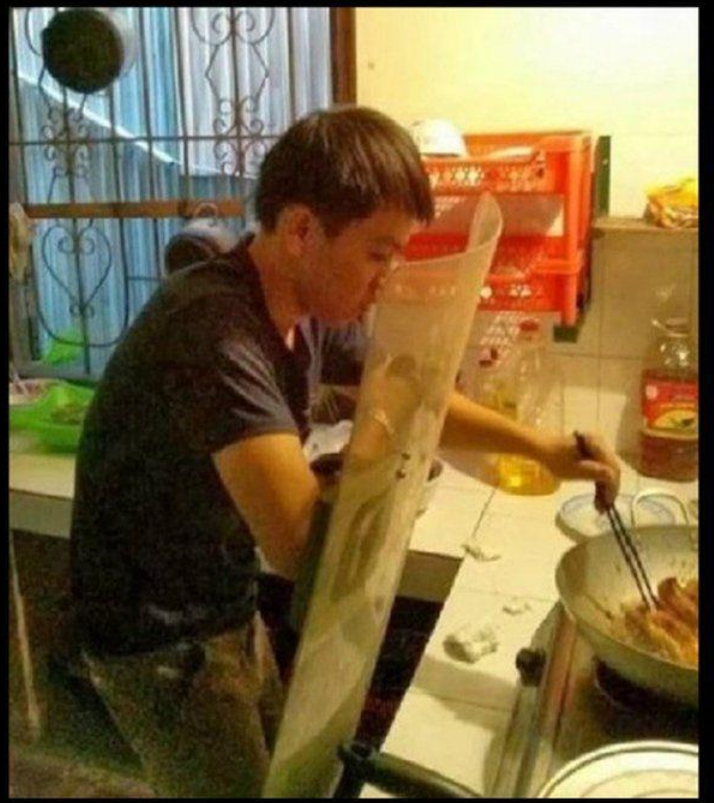
Wróćmy do kuchni

MAN IN THE KITCHEN

EXPECTATIONS



REALITY



Jak być tym, który wie?

A czy tak naprawdę musimy wiedzieć?

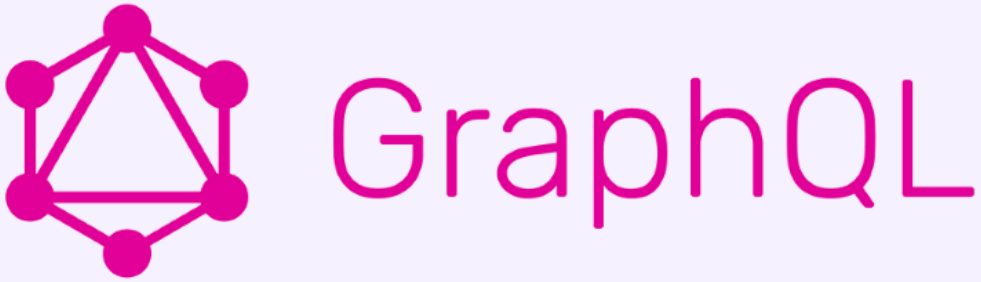
Od czego jest robot kuchenny?

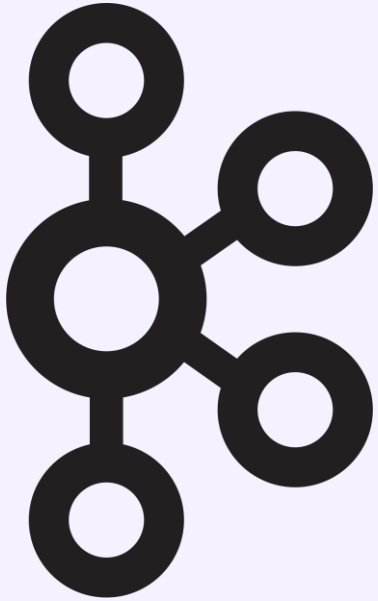
can-i-deploy

```
pact-broker can-i-deploy --participant frontend --version 1 \
--participant backend --version 5
```


{ REST }


gRPC


GraphQL




TS
.NET
Core
Java
GO
JS
php
R

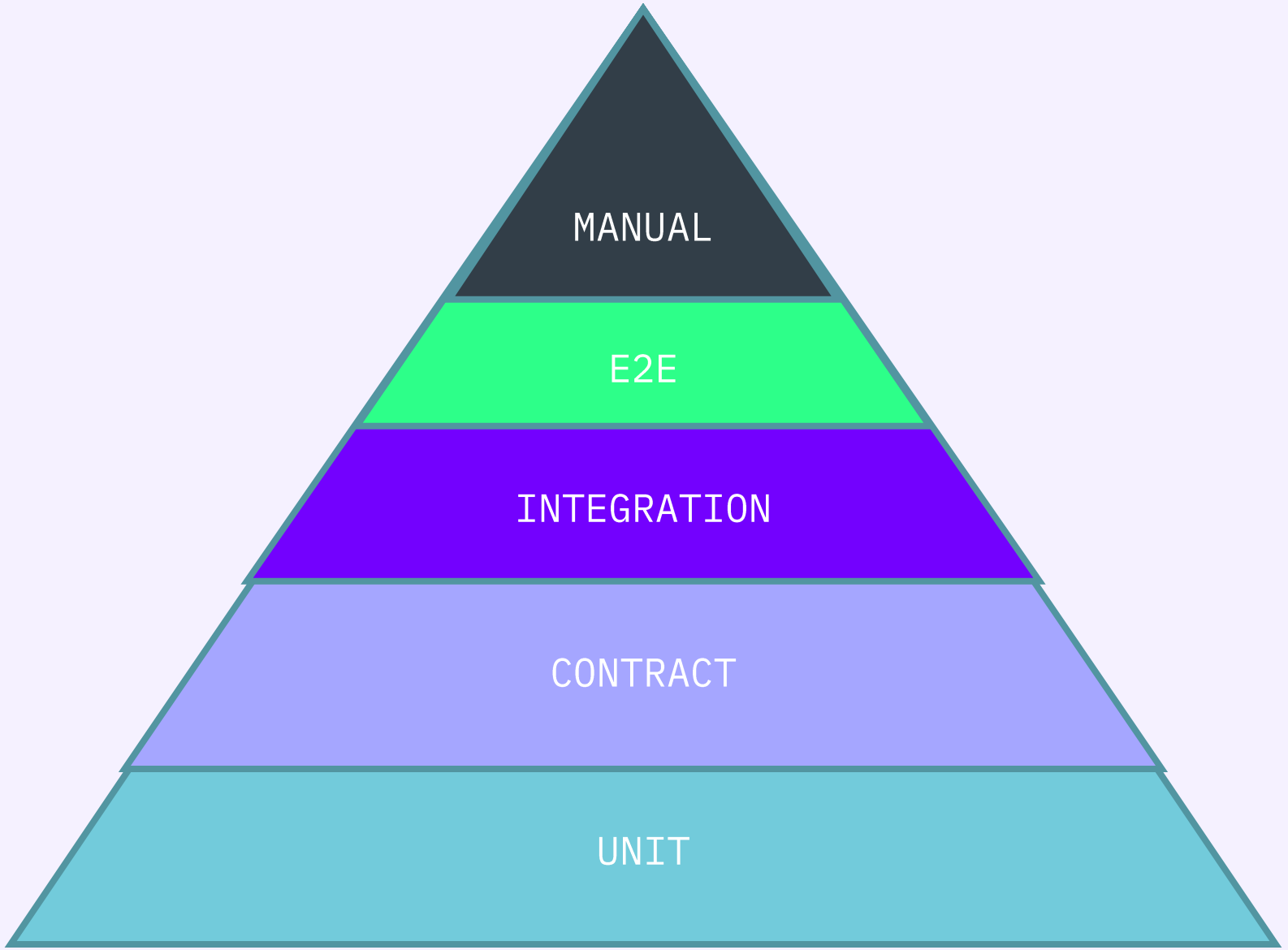
Gdzie jest haczyk?

Zakres mockowania

Początkowa konfiguracja

Najważniejsze jest gadanie


You know me, blah blah blah blah
kimidakewoooo.tumblr.com


MANUAL
E2E
INTEGRATION
CONTRACT
UNIT


R'n'G Kitchen

Dzięki!

pact.io

docs.pact.io

mbaranowski.pl