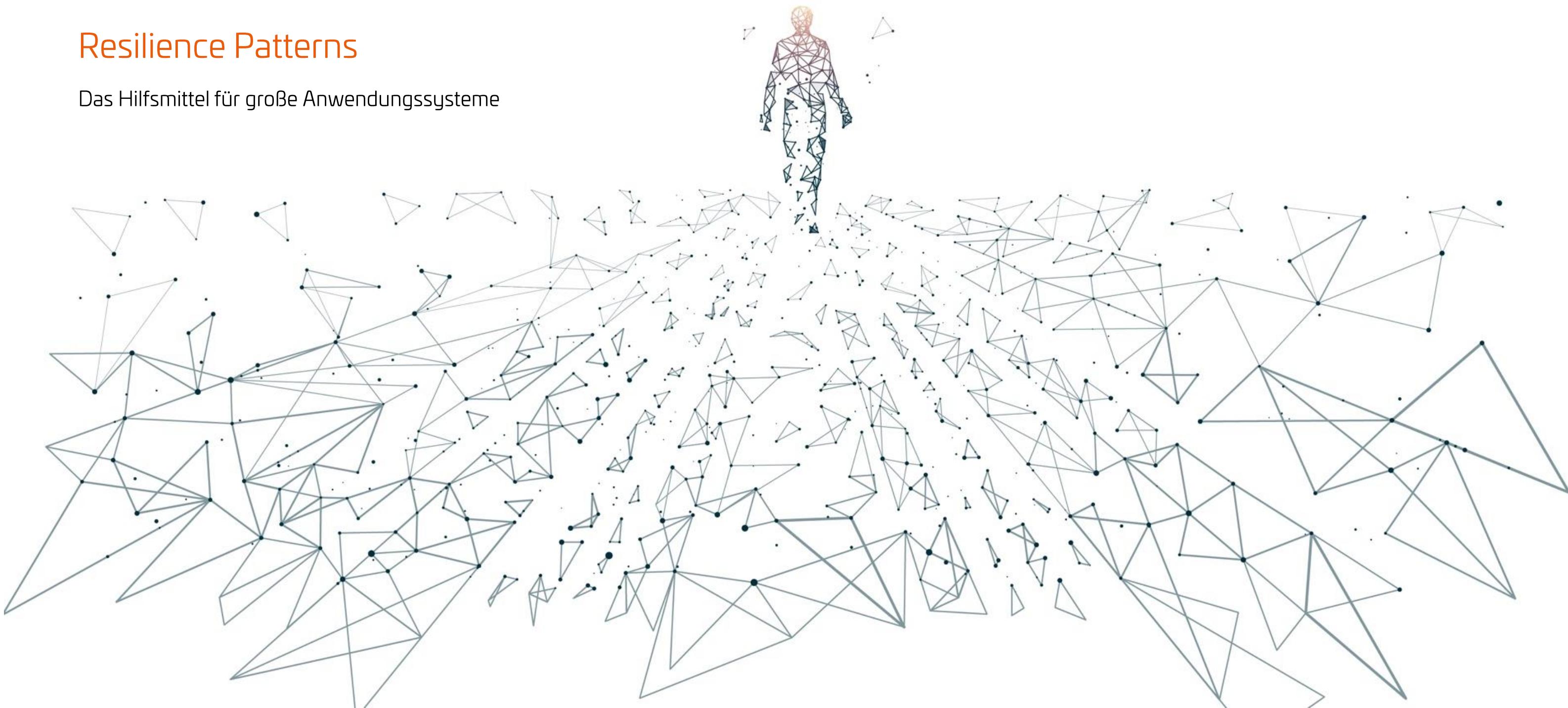
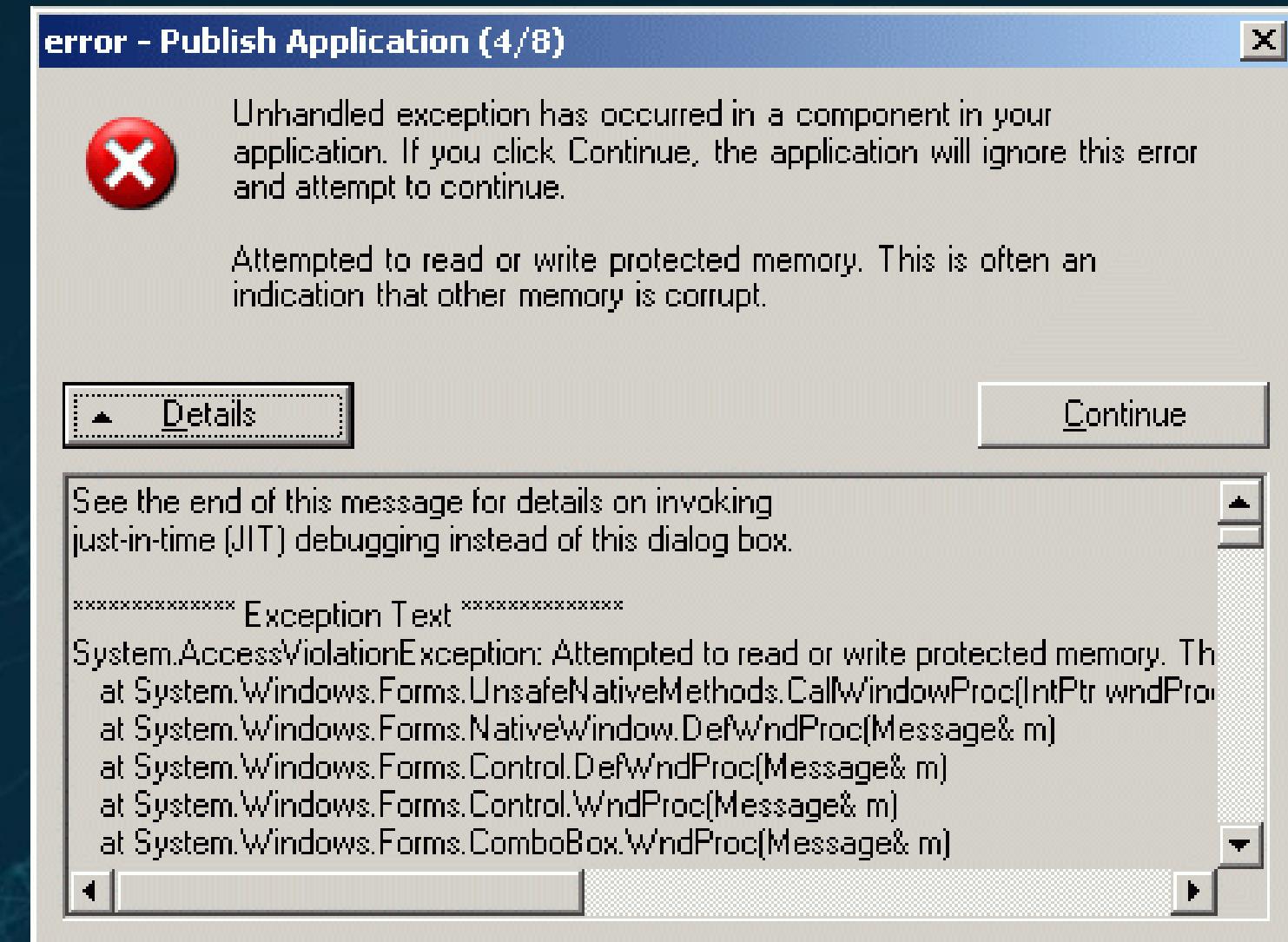


Resilience Patterns

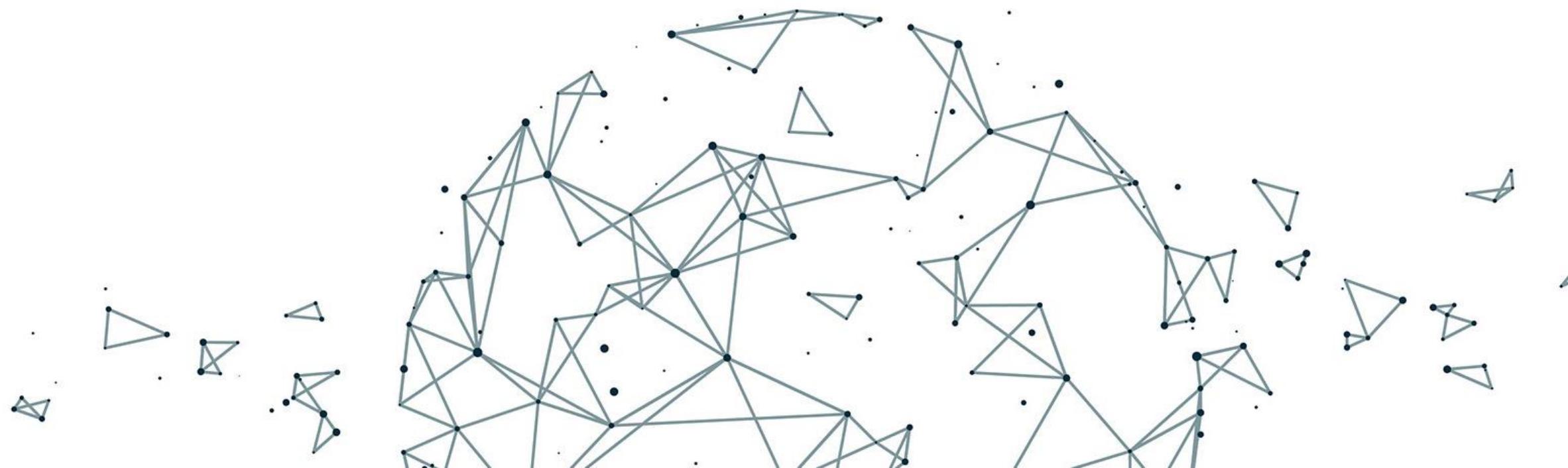
Das Hilfsmittel für große Anwendungssysteme



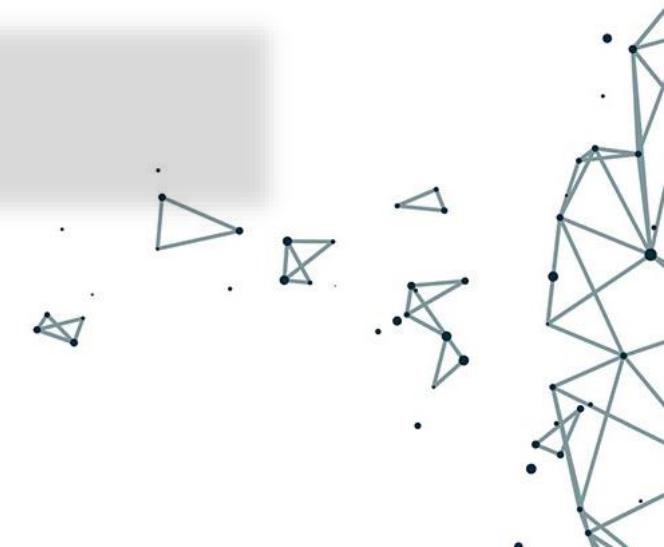
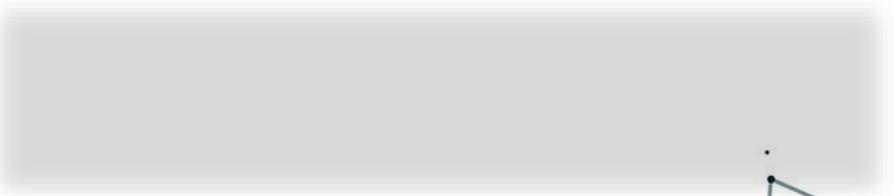


Agenda

- Was ist „Resilience“?
- Pattern
- Demo: Reactive Circuit Breaker



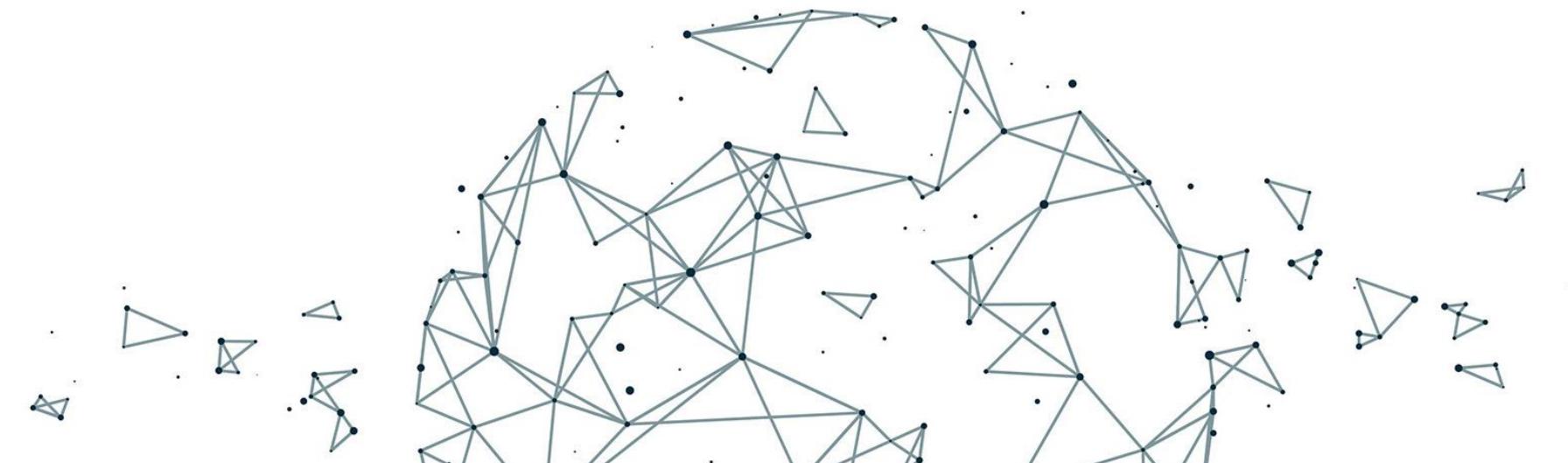
Was ist Resilience?



WIKIPEDIA

„Resilienz bezeichnet ... die Fähigkeit von technischen Systemen, bei Störungen bzw. Teil-Ausfällen nicht vollständig zu versagen, sondern wesentliche Systemdienstleistungen aufrechtzuerhalten.“

[https://de.wikipedia.org/wiki/Resilienz_\(Ingenieurwissenschaften\)](https://de.wikipedia.org/wiki/Resilienz_(Ingenieurwissenschaften))



Erwartungen

- Hohe Verfügbarkeit des Systems
- Schnelle Antwortzeiten
- Probleme sollen dem Endbenutzer nicht (negativ) auffallen





Verteilte Systeme

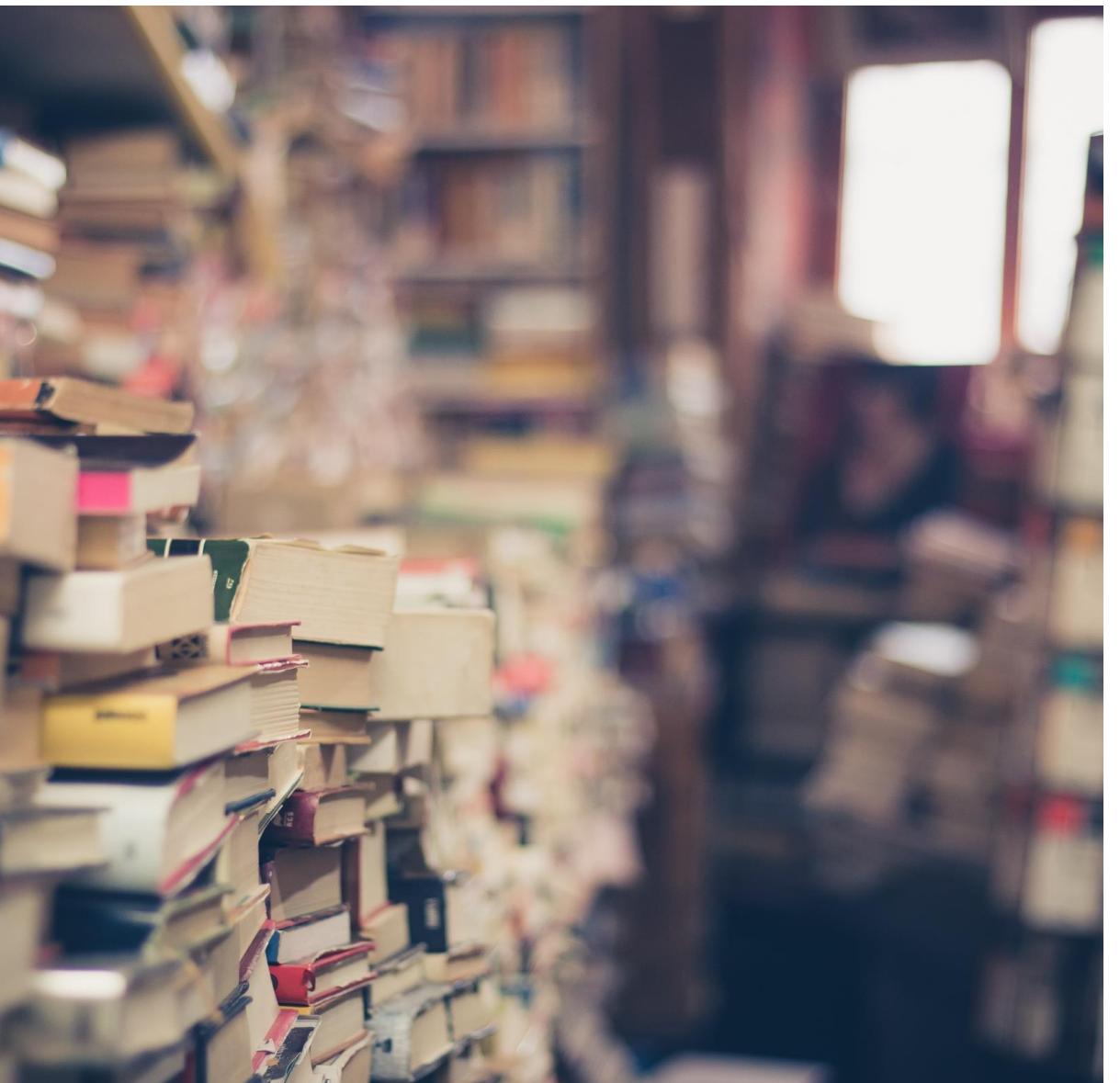
- ▷ Komplexe Systeme
- ▷ Abhängigkeiten untereinander
- ▷ Kommunikation über Netzwerk
- ▷ Netzwerk ist nicht so konsistent wie angenommen
- ▷ Fehler passieren und sind nicht die Ausnahme





Pattern Sammlung

- ▷ Die wichtigsten Resilience Patterns in einer Kurzübersicht



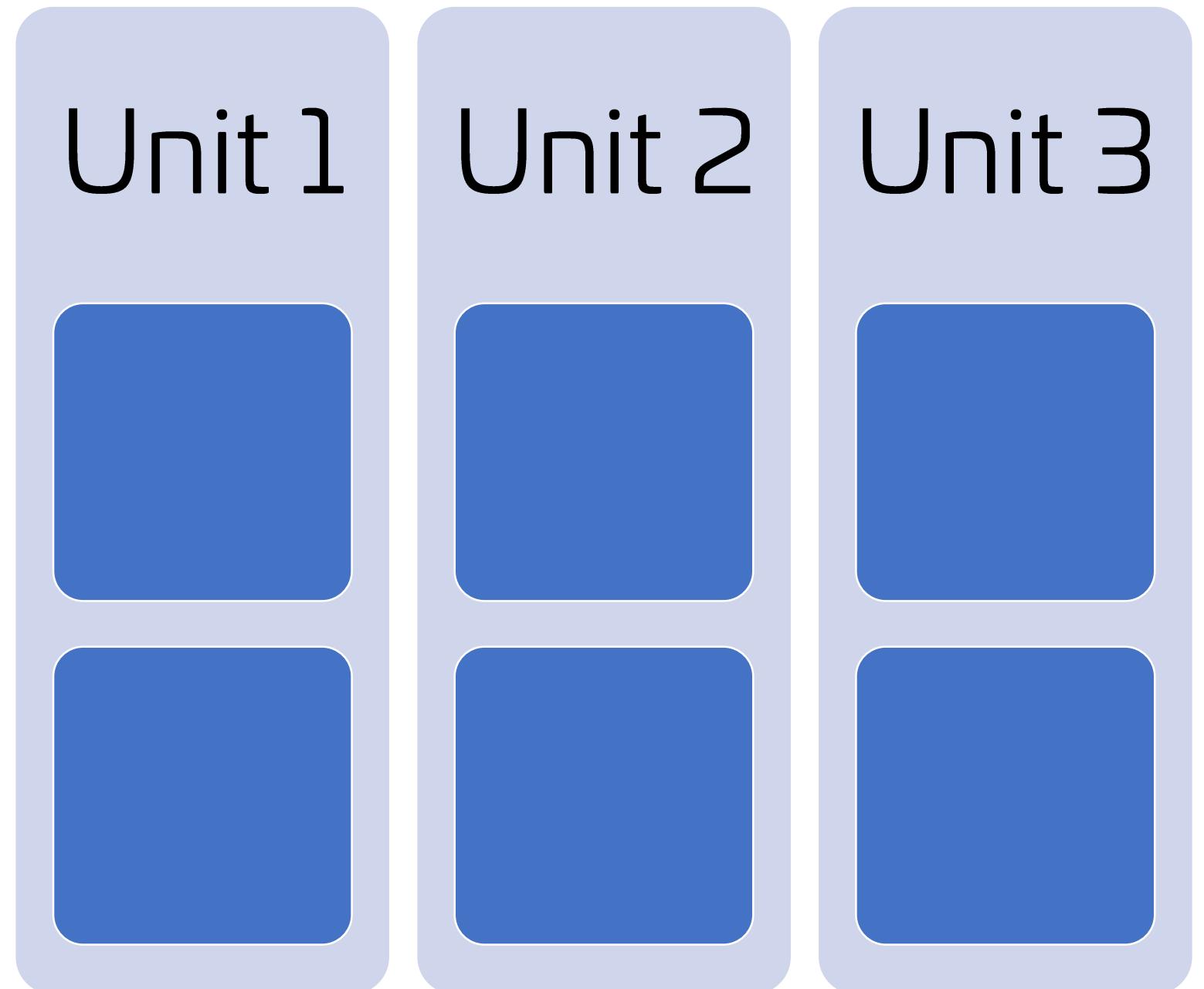
Isolation

- ▷ Aufteilen der Anwendung
- ▷ Isolieren der Anwendungsteile voneinander
- ▷ Fehlerkaskaden vermeiden

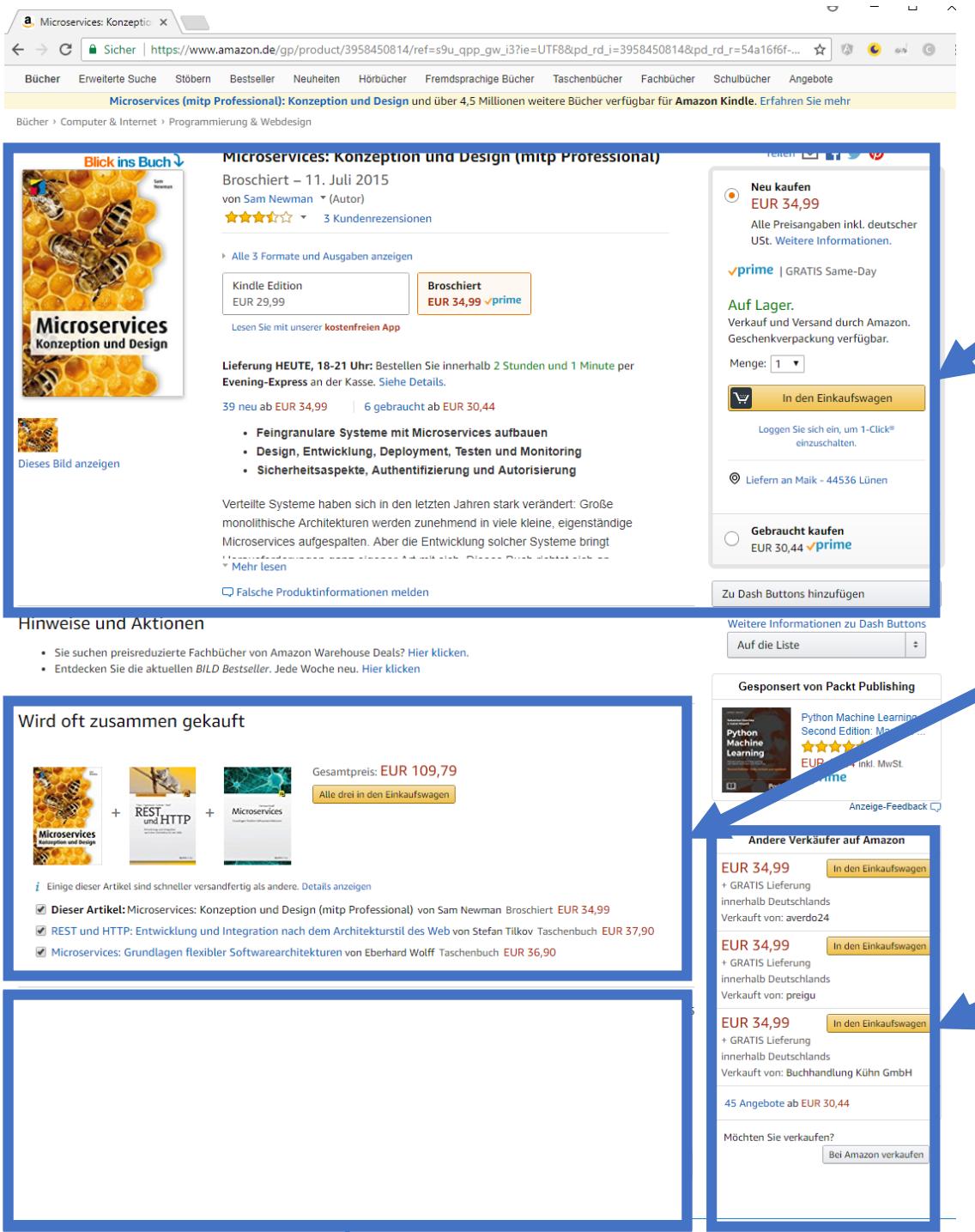


Bulkhead

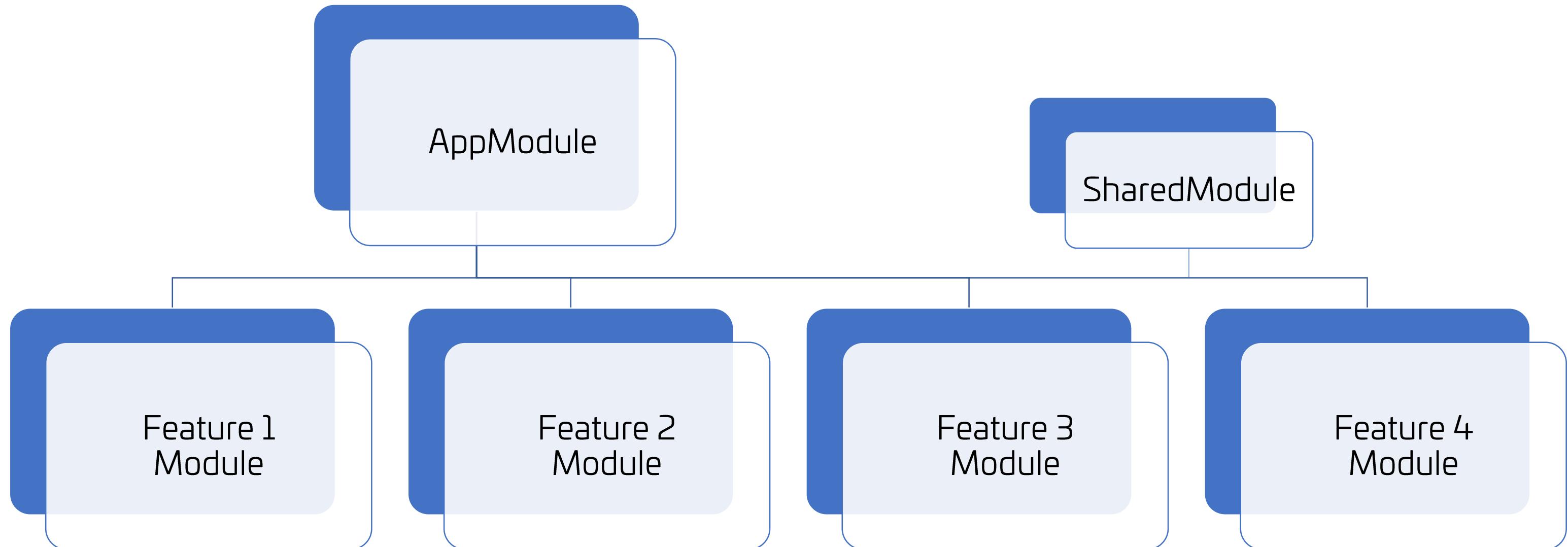
- ▷ Zentrales Isolations-Pattern
 - ▷ „Failure Units“
 - ▷ „Units of Migration“
- ▷ Unit als Skalierungselement
- ▷ Unit als Redundanz



Bulkhead

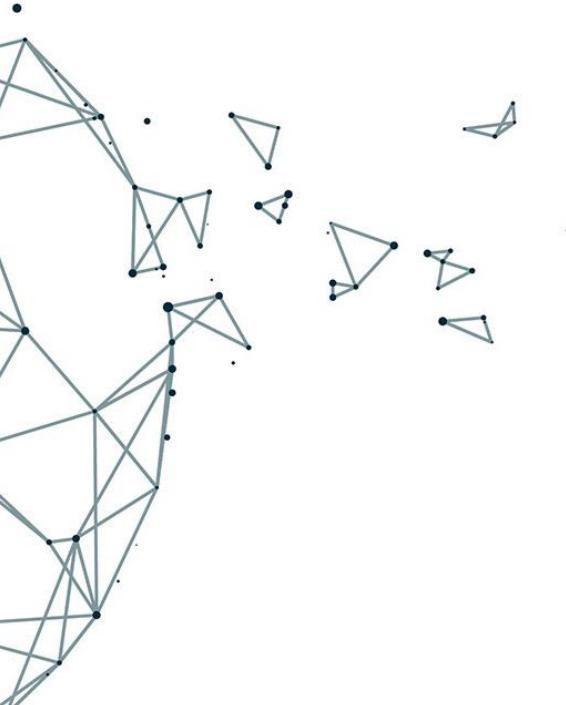


Bulkhead In Angular



Lose Kopplung (Loose Coupling)

- ▷ Autarke Systeme sind sehr selten
- ▷ Aktionen beeinflussen andere Systeme
- ▷ Möglichst lose Kopplung zwischen Systemen

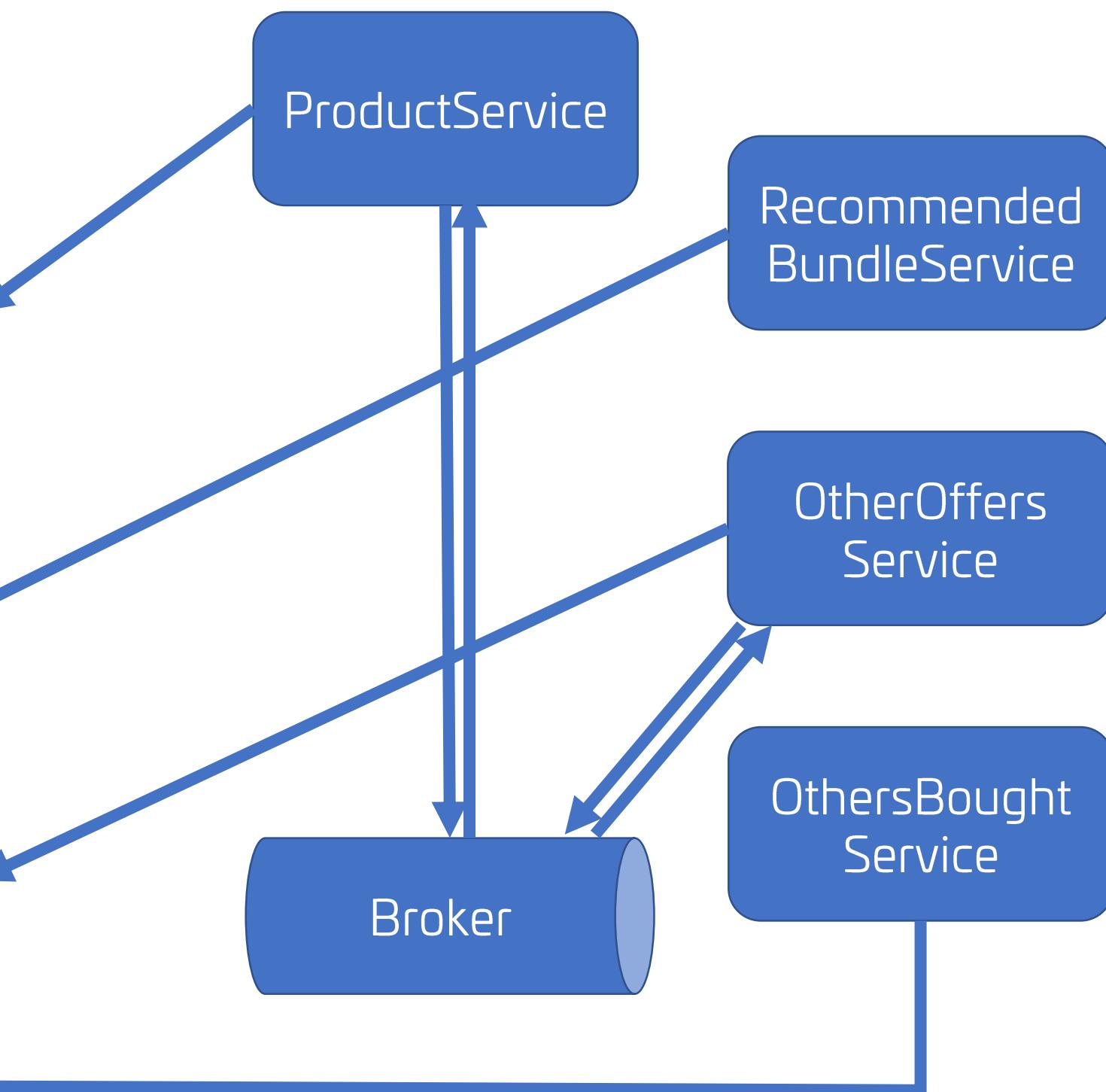
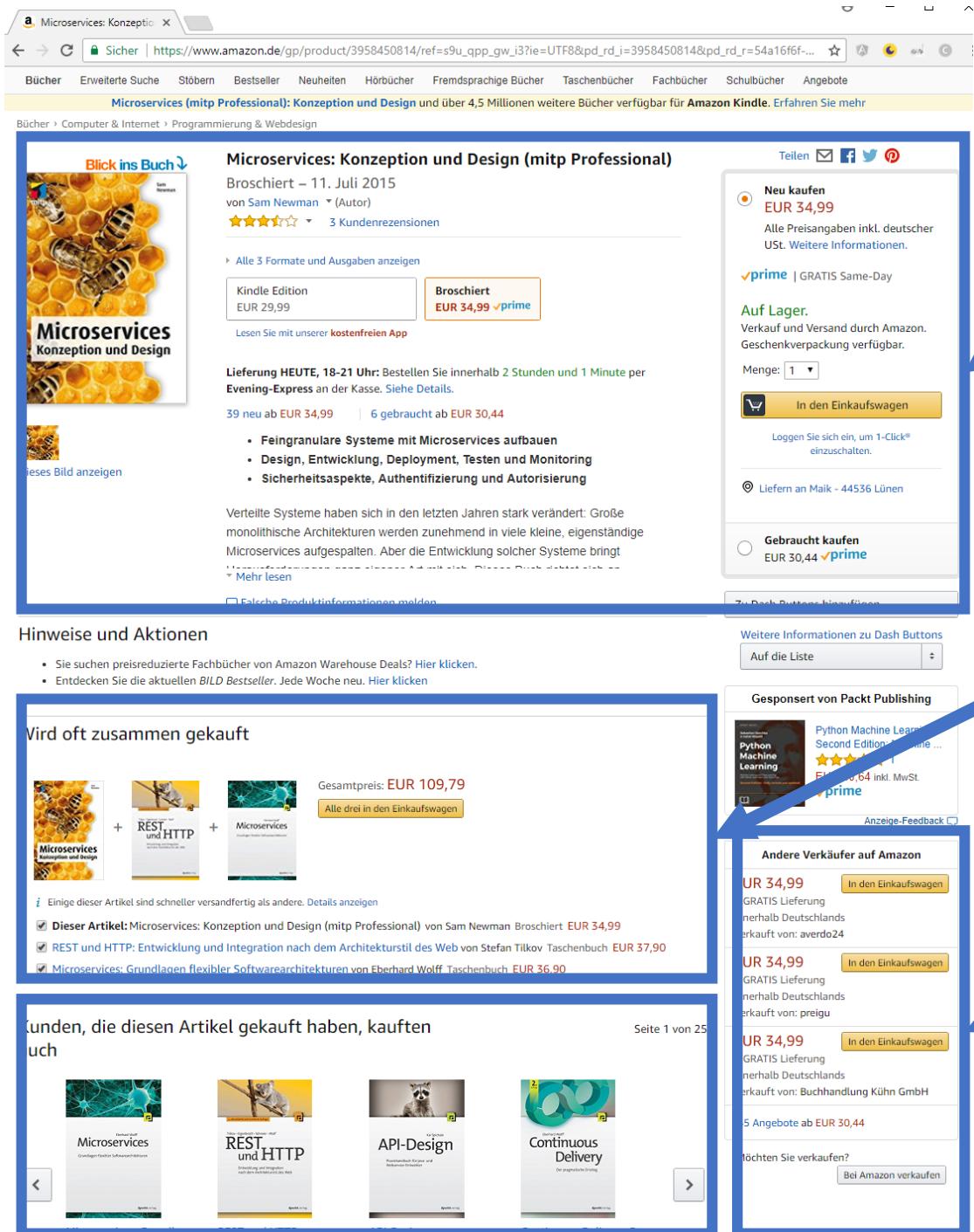


Asynchrone Kommunikation

- Trennung von Sender und Empfänger
- Keine direkte Antwort erwartet
- Entspannt zeitliche Abhängigkeiten
- „Eventual Consistency“



Asynchrone Kommunikation



Asynchrone Kommunikation In Angular

- RxJS ist die Grundlage
- Alles bereits mit an Board
- Subject als einfacher „MessageBus“
- Shared Service

```
@Injectable()
class MessageBus {

    private messageBus: Rx.Subject;

    constructor() {
        this.messageBus = new Rx.Subject();
    }

    subscribe<T>(action: (value:T) => void) {
        this.messageBus.subscribe(action);
    }

    send<T>(value: T) {
        this.messageBus.next(value);
    }
}
```

Latenz Kontrolle

- ▷ Erkennen und Behandeln von zu langen Antwortzeiten
- ▷ Summe von zu großen Latenzen problematisch
- ▷ Vordefinierte Timeouts
- ▷ „Fail Fast“



Timeouts in Angular

- ▷ HttpClient nutzt RxJS
- ▷ Direkt verfügbar über RxJS Operator

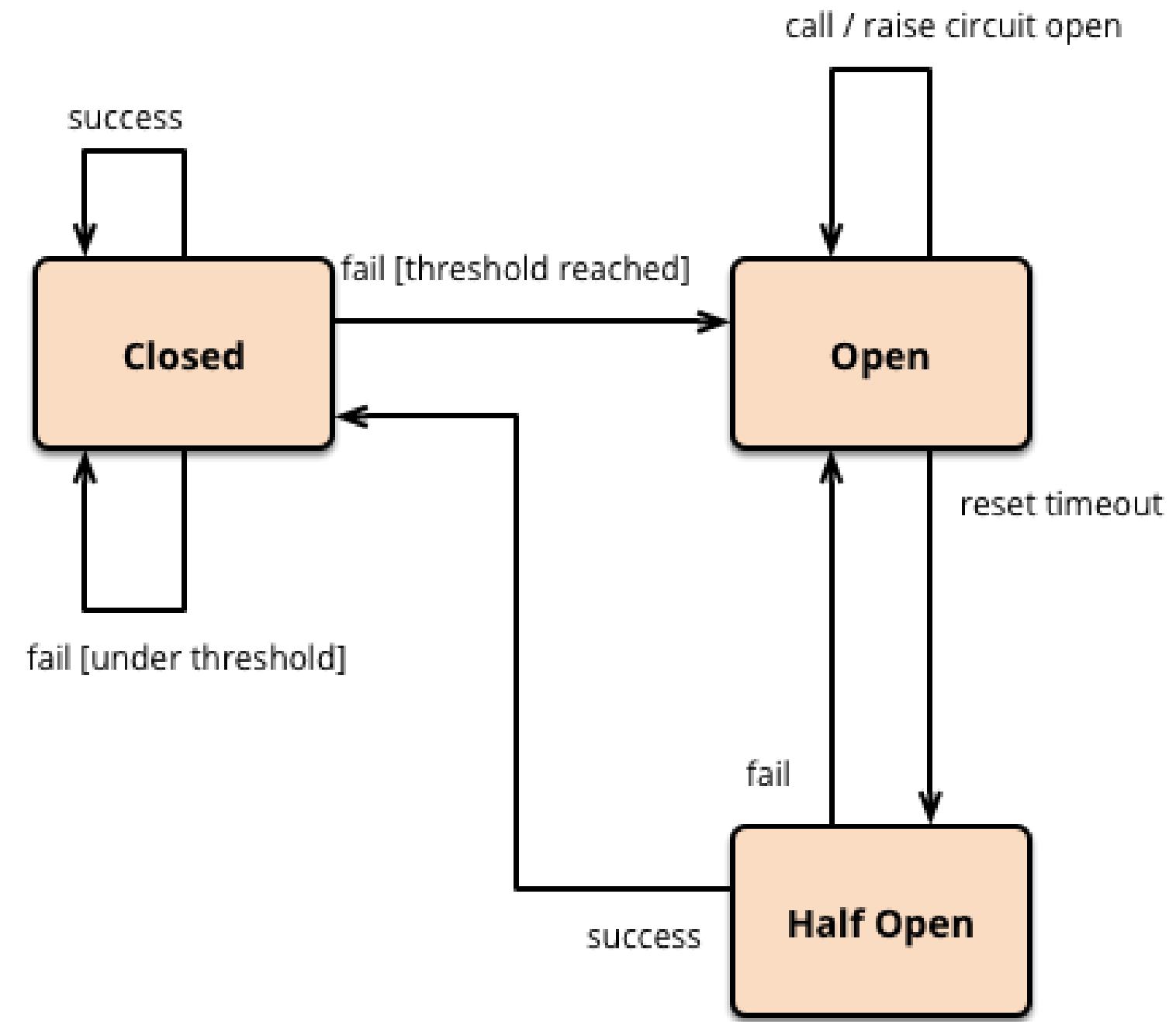
```
@Injectable()
class MyService {

  constructor(private http: HttpClient) {}

  getData() {
    return this.http.get("/api/data")
      .pipe(
        timeout(3000) // in ms
      )
  }
}
```

Circuit Breaker

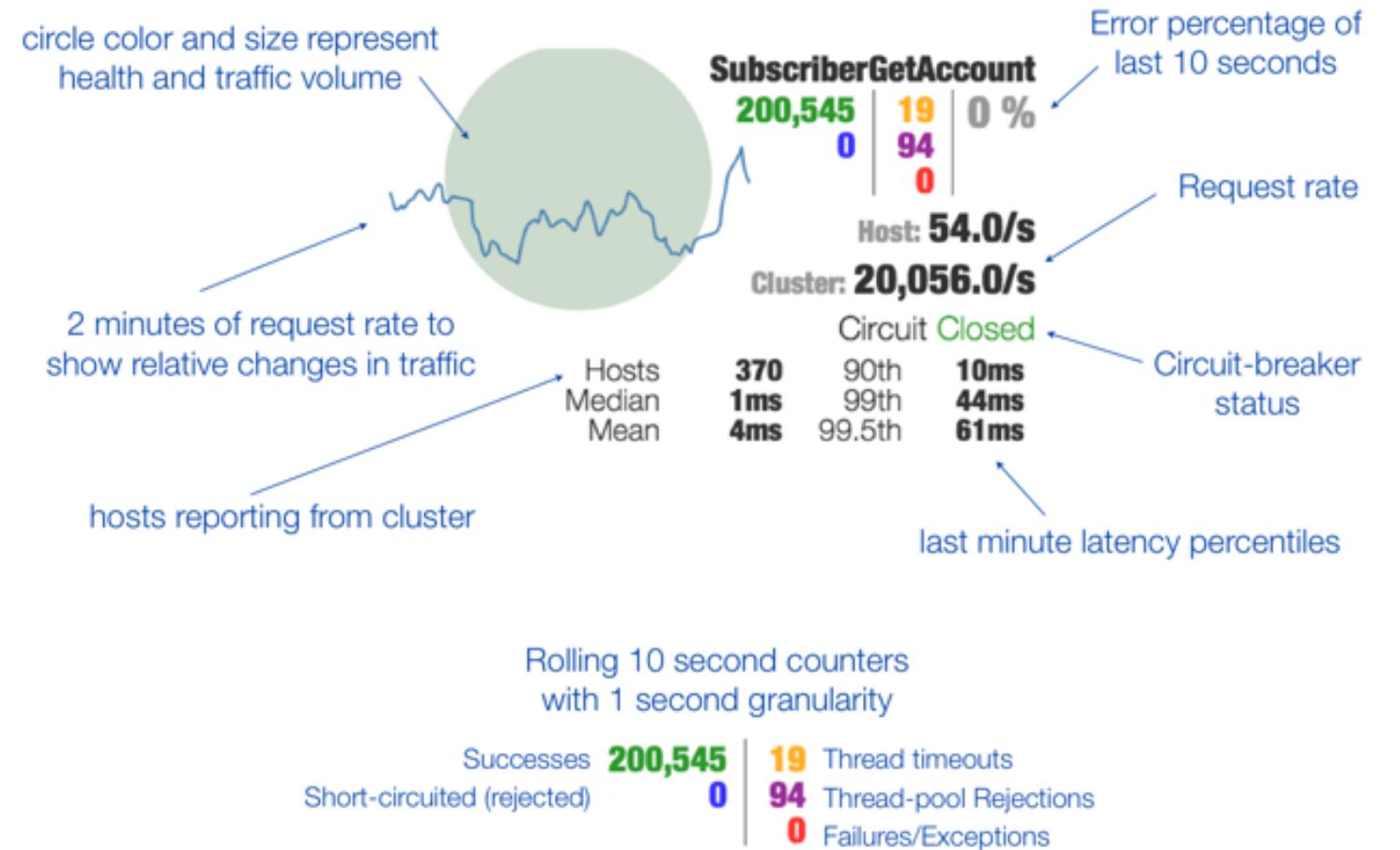
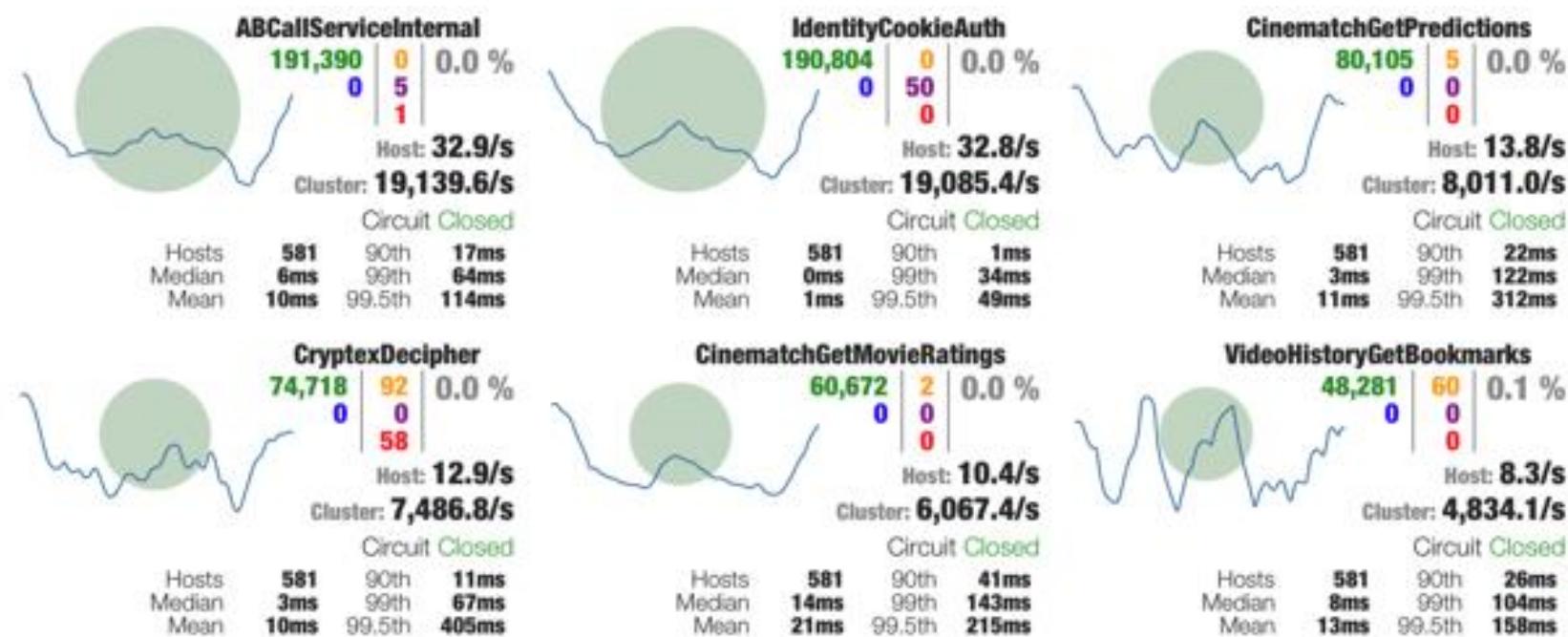
- „Sicherung“ um die unterliegende Ressource zu schützen
- Lässt eine bestimmte Anzahl von Fehlern zu
- Weitere Requests werden unterbunden
- Nach definierter Zeit wird erneut versucht



```
<nav id="nav" role="navigation">
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="home-events.html">Home Events</a>
    <li><a href="multi-col-menu.html">Multiple Co
    <li class="has-children"> <a href="#" class="c
      <ul>
        <li><a href="tall-button-header.html">
        <li><a href="image-logo.html">Image Lo
        <li class="active"><a href="tall-logo.
      .
```

Code Demo: Reactive Circuit Breaker

Circuit Breaker @ Netflix



Fazit

- Verteilte Systeme sind überall
- Fehler passieren und sind nicht vorhersehbar
- Resilience Patterns helfen



Links

- ▷ Patterns of Resilience (Uwe Friedrichsen)

<https://de.slideshare.net/ufried/patterns-of-resilience>

- ▷ Netflix TechBlog

<https://medium.com/netflix-techblog>

- ▷ Hystrix

<https://medium.com/netflix-techblog/introducing-hystrix-for-resilience-engineering-13531c1ab362>



Ich freue mich auf euer Feedback und einen regen Austausch mit euch!

Kontaktdaten



Maik Schöneich
Lead Software Engineer

E-Mail: msh@maximago.de
Mobil: +49 171 55 00 878

<https://www.maximago.de/>
<https://www.facebook.com/maximago/>