



# ArchUnit

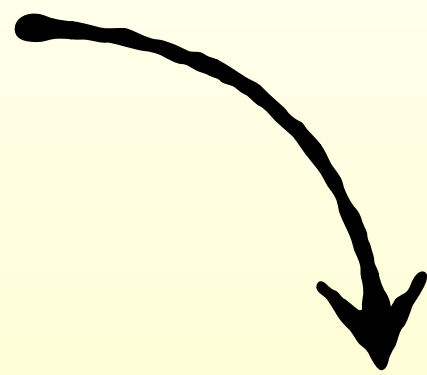
Testen von Architektur und Design

Thomas Much

 @thmuch

25.01.2022

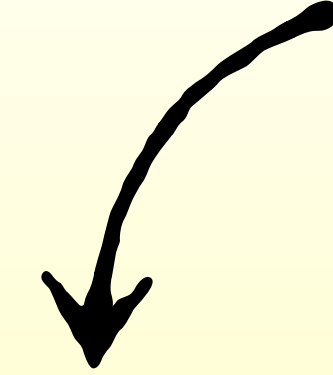
**Autor**



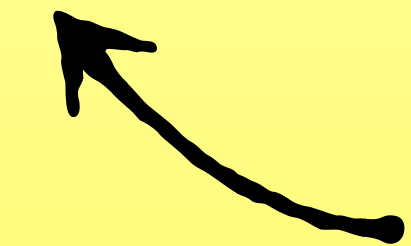
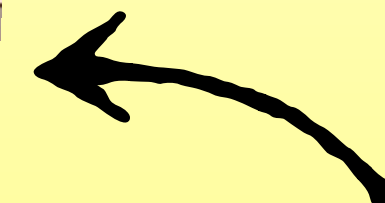
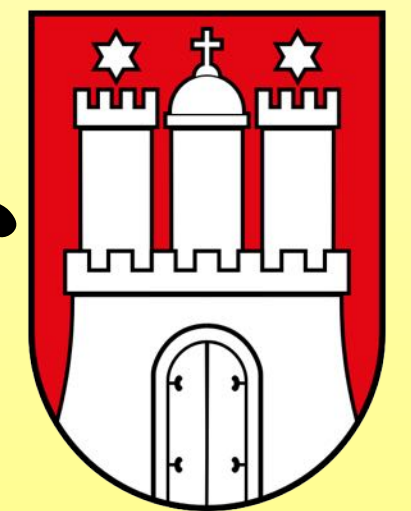
Peter Gafert  
@codecholeric  
@archtests

zahlreiche  
„Contributors“

**glücklicher Anwender**



Thomas Much  
@thmuch



# Was ist Architektur?

Entscheidungen, Technologien ...  
alles „Wichtige“, was schwer zu ändern ist

Struktur und Konventionen innerhalb eines Artefakts (Services)

Kommunikation zwischen den (Teil-)Systemen

Gemeinsames Verständnis über das System und seine Teile

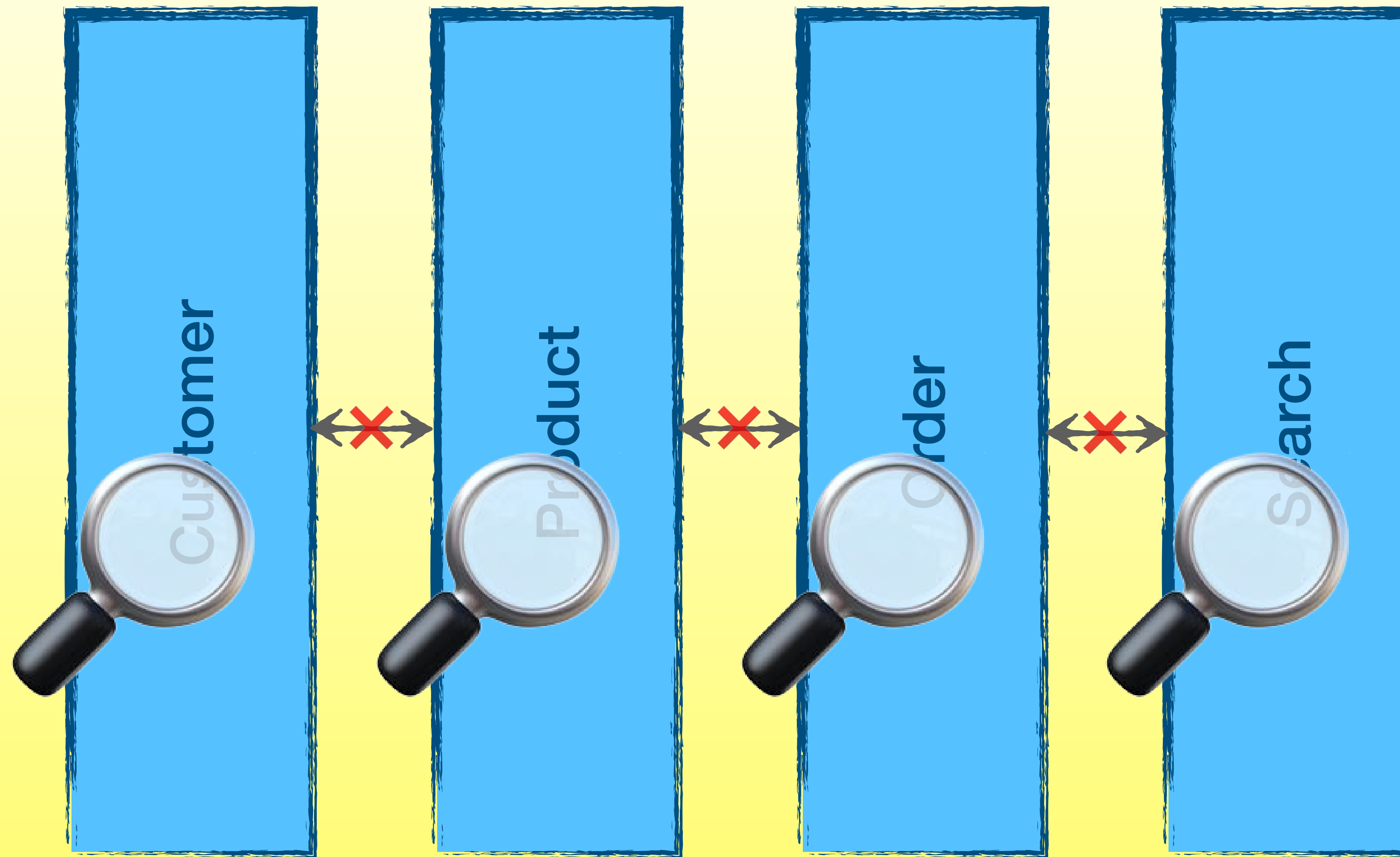
...

Thema heute:  
**ArchUnit**

Architektur und Design automatisiert prüfen –

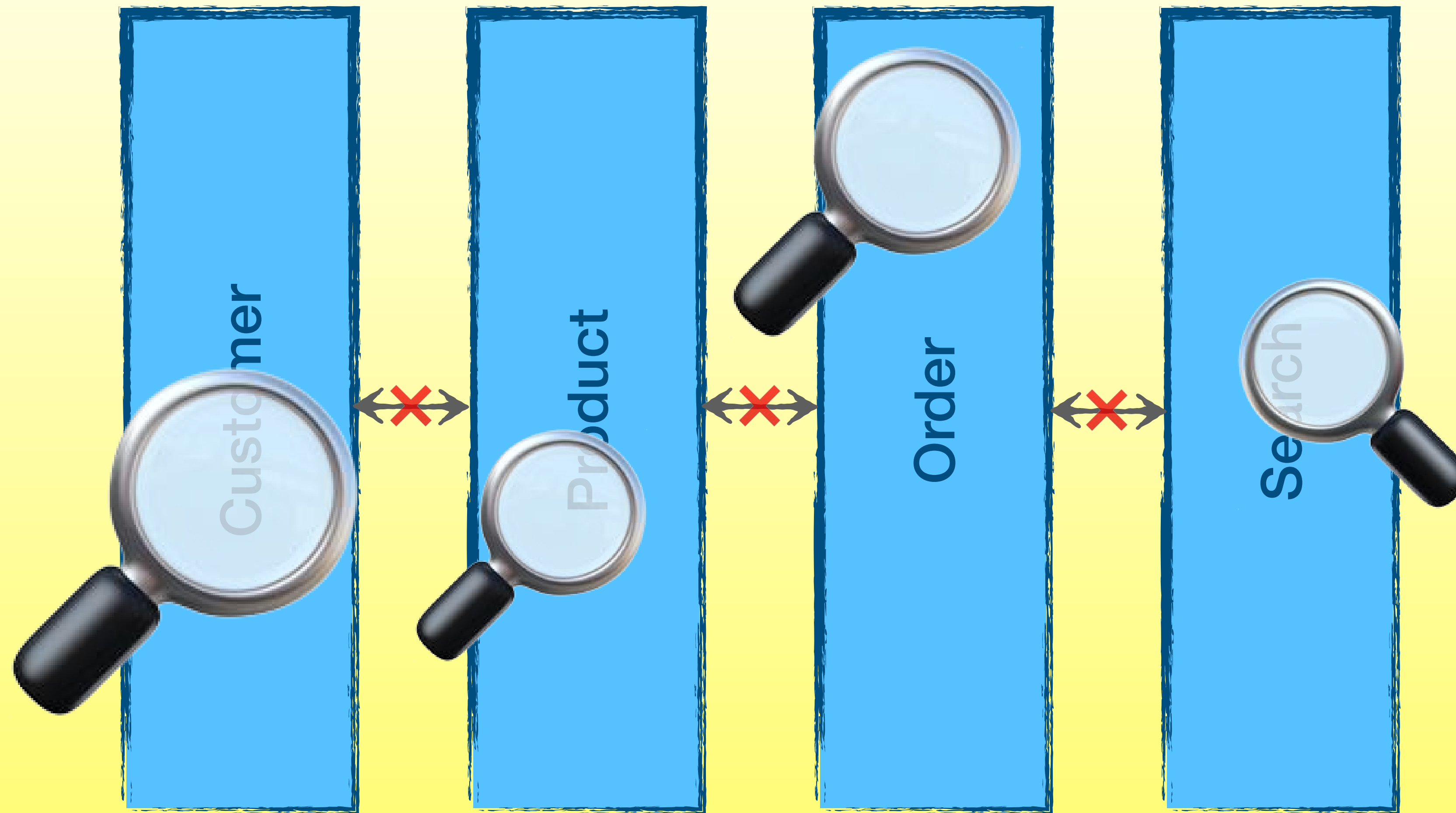
**Warum? Was genau?**

# Microservices & SCS\*



\*) „Self-Contained Systems“, siehe z.B. <https://scs-architecture.org/>

# Microservices & SCS\*

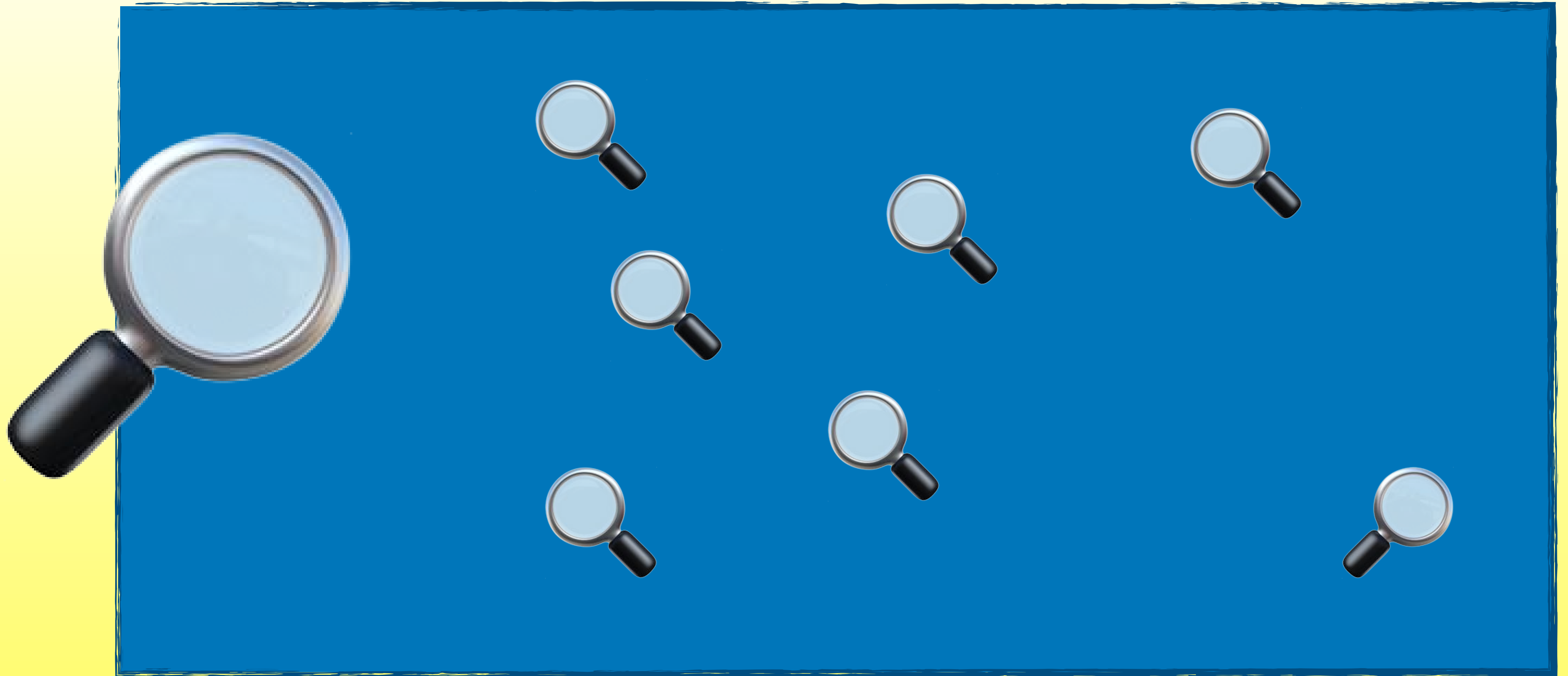


\*) „Self-Contained Systems“, siehe z.B. <https://scs-architecture.org/>

# Monolithen

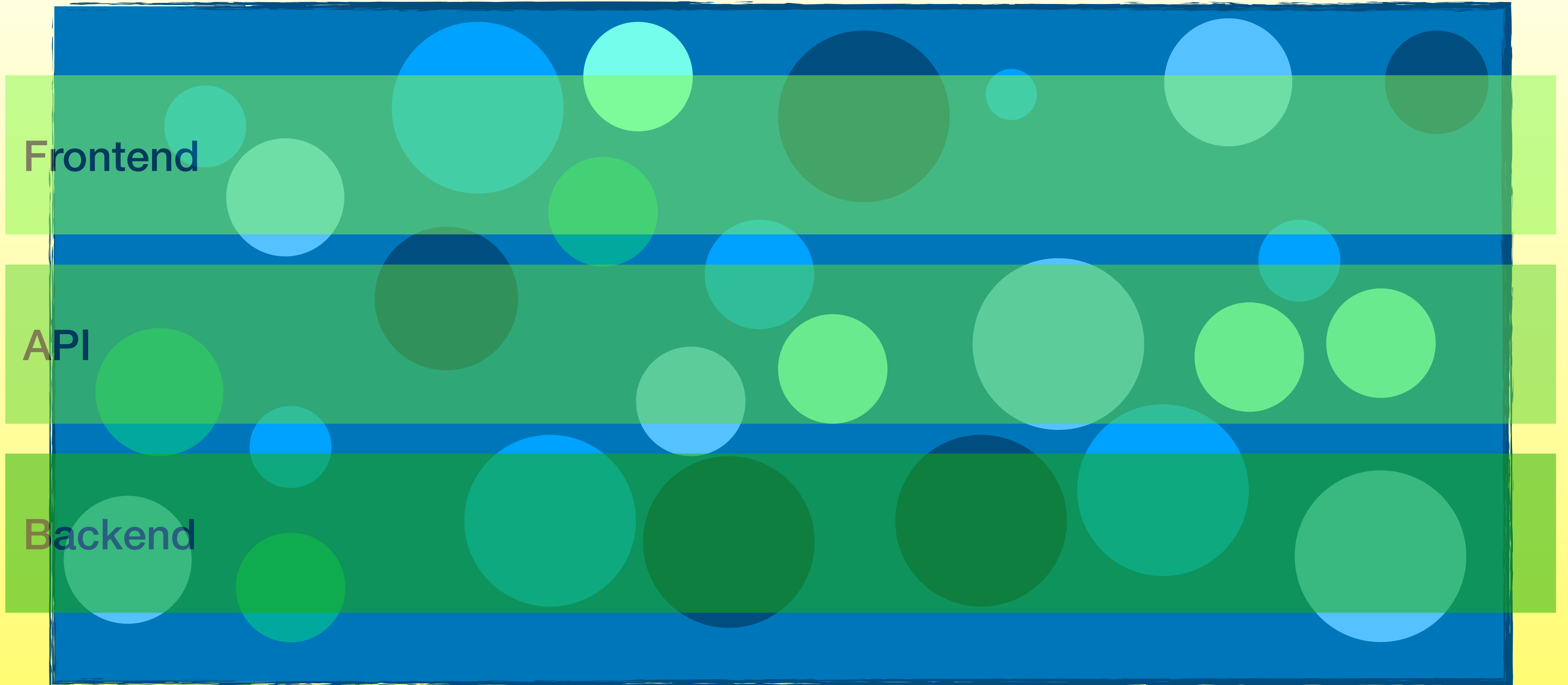


# Monolithen

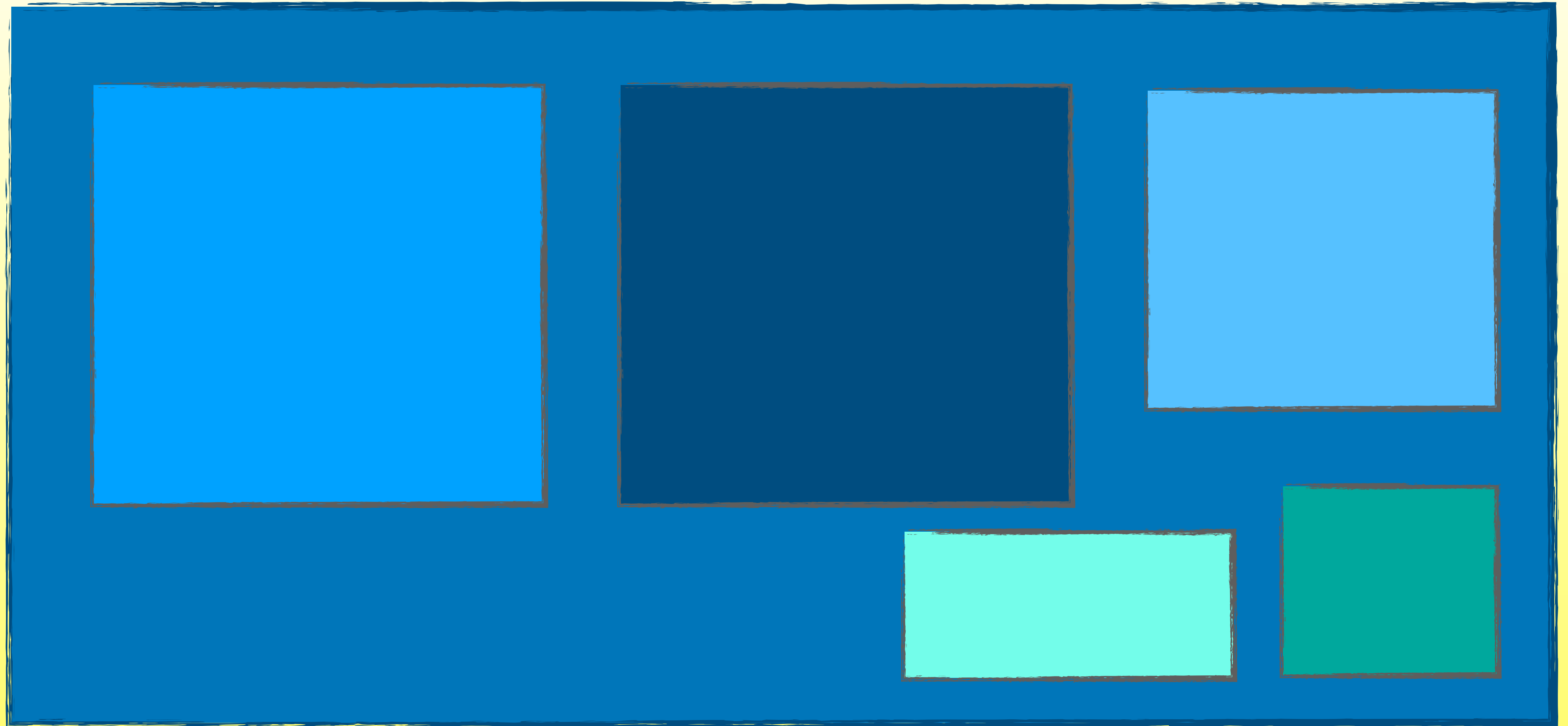




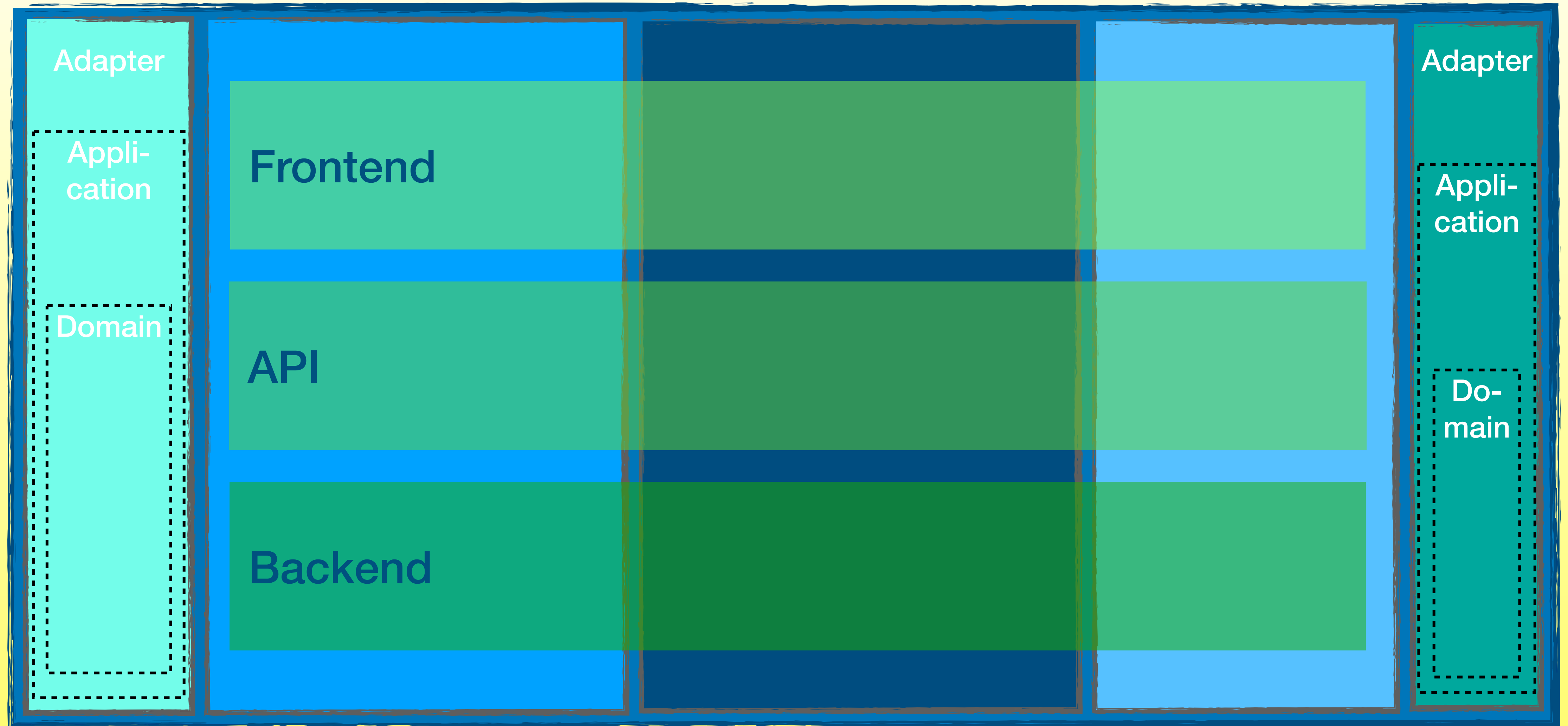
# Monolithen



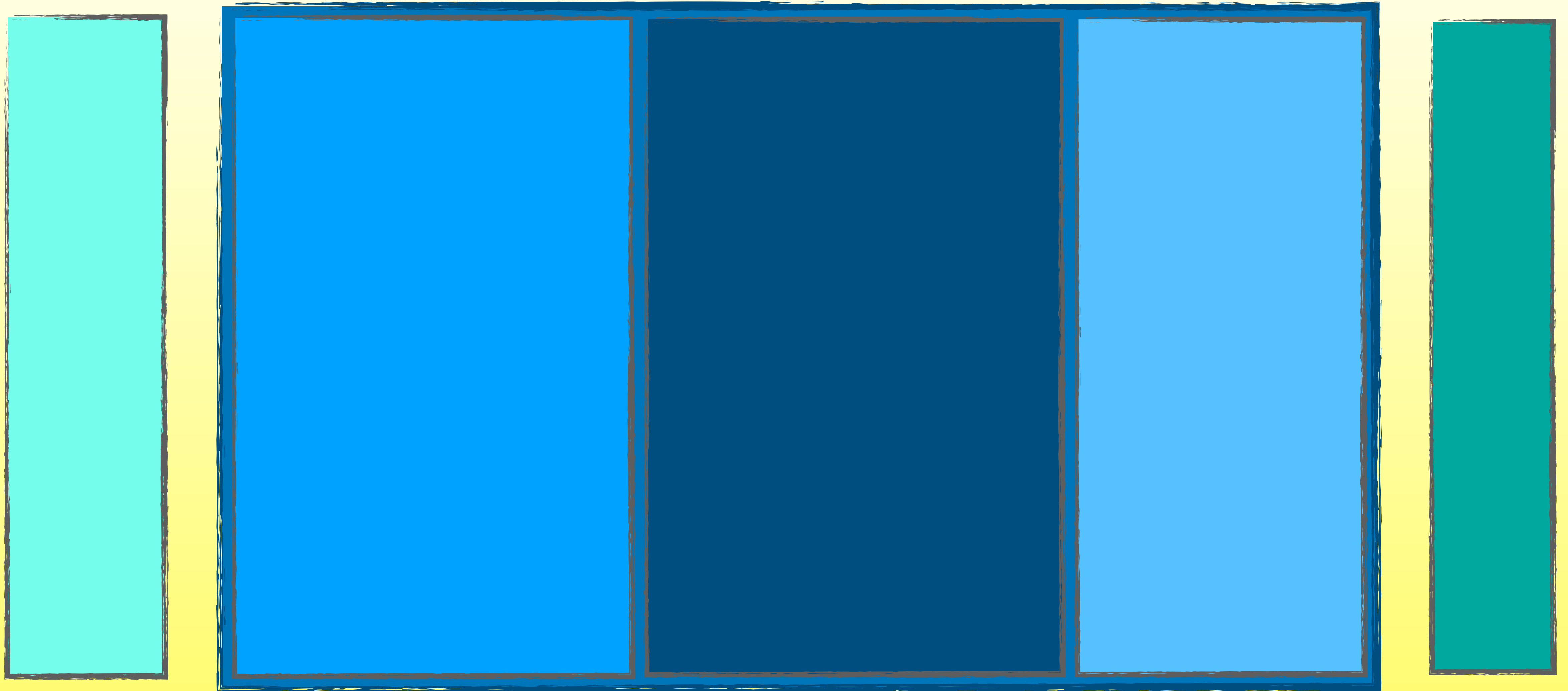
# Freundliche Monolithen



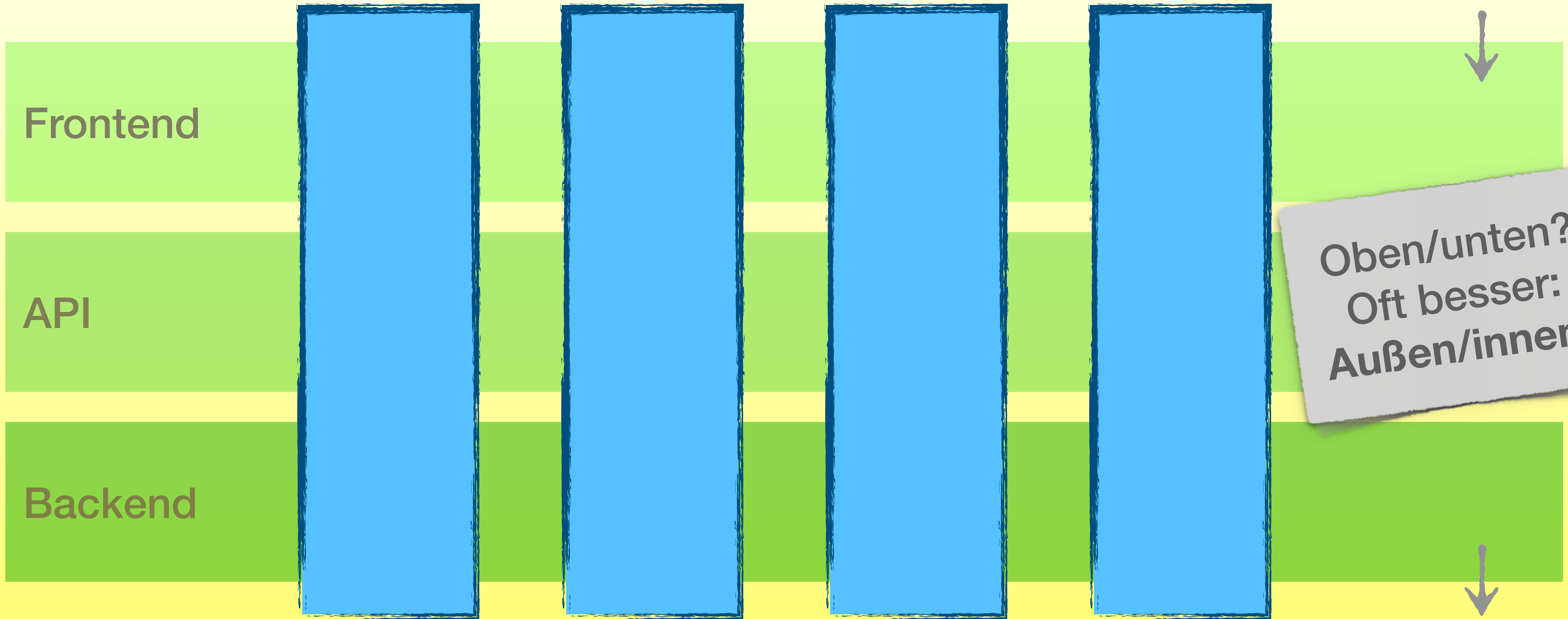
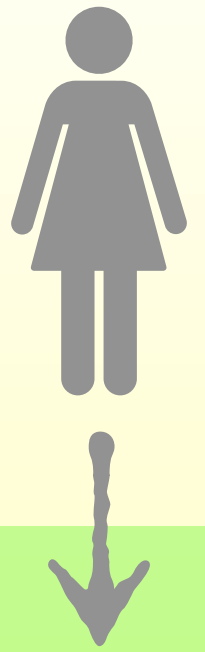
# Noch freundlichere Monolithen



# Abgespeckte Monolithen



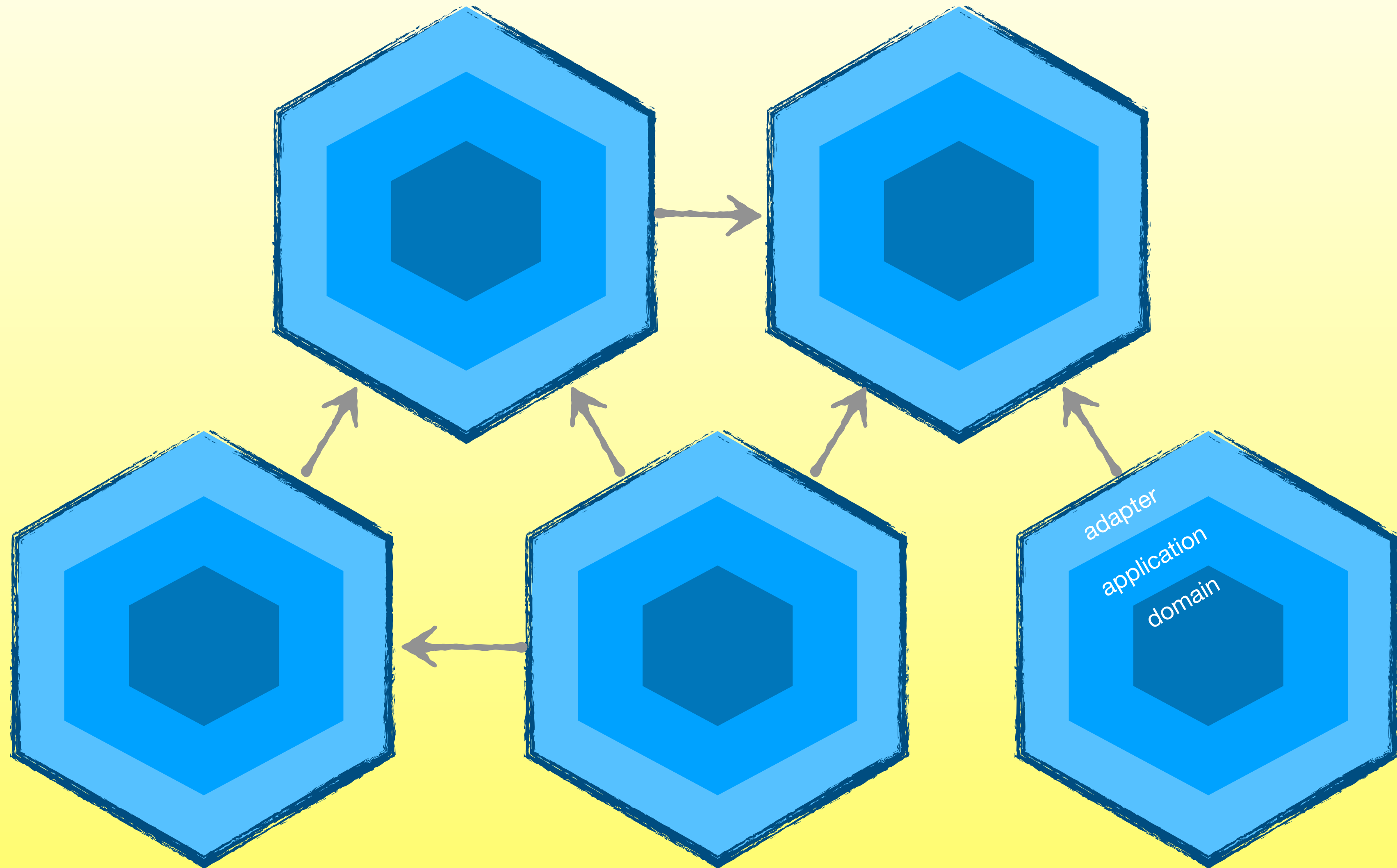
# Was prüfen? z.B. Schichten



Oben/unten?  
Oft besser:  
Außen/innen!



# Was prüfen? z.B. Ports & Adapters



# Architektur testen – warum?

Wartbarkeit  
Ersetzbarkeit  
Dokumentation  
Kommunikation  
Verständlichkeit  
Code-Conventions

Wir  
sind  
ein  
Team!

Wir sind ein Unternehmen!  
Mit vielen Teams,  
über viele Jahre

# Architektur testen

Abhängigkeiten, Kohäsion & Kopplung

Konventionen & Patterns



# Mein Weg zu ArchUnit



**Ende 2017**  
**„ArchUnit 0.4“**

Classycle

Checkstyle

möglichst einfach

~~kommerziell~~



Architektur-Prüfungen als **Unit-Tests**

Normaler **Java-Code!**

**Flexibel erweiterbar** – auch Design-Prüfungen realisierbar

Prüfung auf **Bytecode-Ebene**

# ArchUnit einbinden

Group ID	Artifact ID	Latest Version		Updated
com.tngtech.archunit	archunit-junit	0.8.3	(6)	20-Jul-2018
com.tngtech.archunit	archunit-junit5-engine-api	0.22.0	(23)	29-Oct-2021
com.tngtech.archunit	archunit-junit5-engine	0.22.0	(23)	29-Oct-2021
com.tngtech.archunit	archunit-junit5-api	0.22.0	(23)	29-Oct-2021
com.tngtech.archunit	archunit-junit5	0.22.0	(14)	29-Oct-2021
com.tngtech.archunit	archunit-junit4	0.22.0	(23)	29-Oct-2021
com.tngtech.archunit	archunit	0.22.0	(29)	29-Oct-2021

```
<dependency>  
  <groupId>com.tngtech.archunit</groupId>  
  <artifactId>archunit-junit5</artifactId>  
  <version>0.22.0</version>  
  <scope>test</scope>  
</dependency>
```

# ArchUnit ausführen

Mit jedem Unit-Test-Framework nutzbar

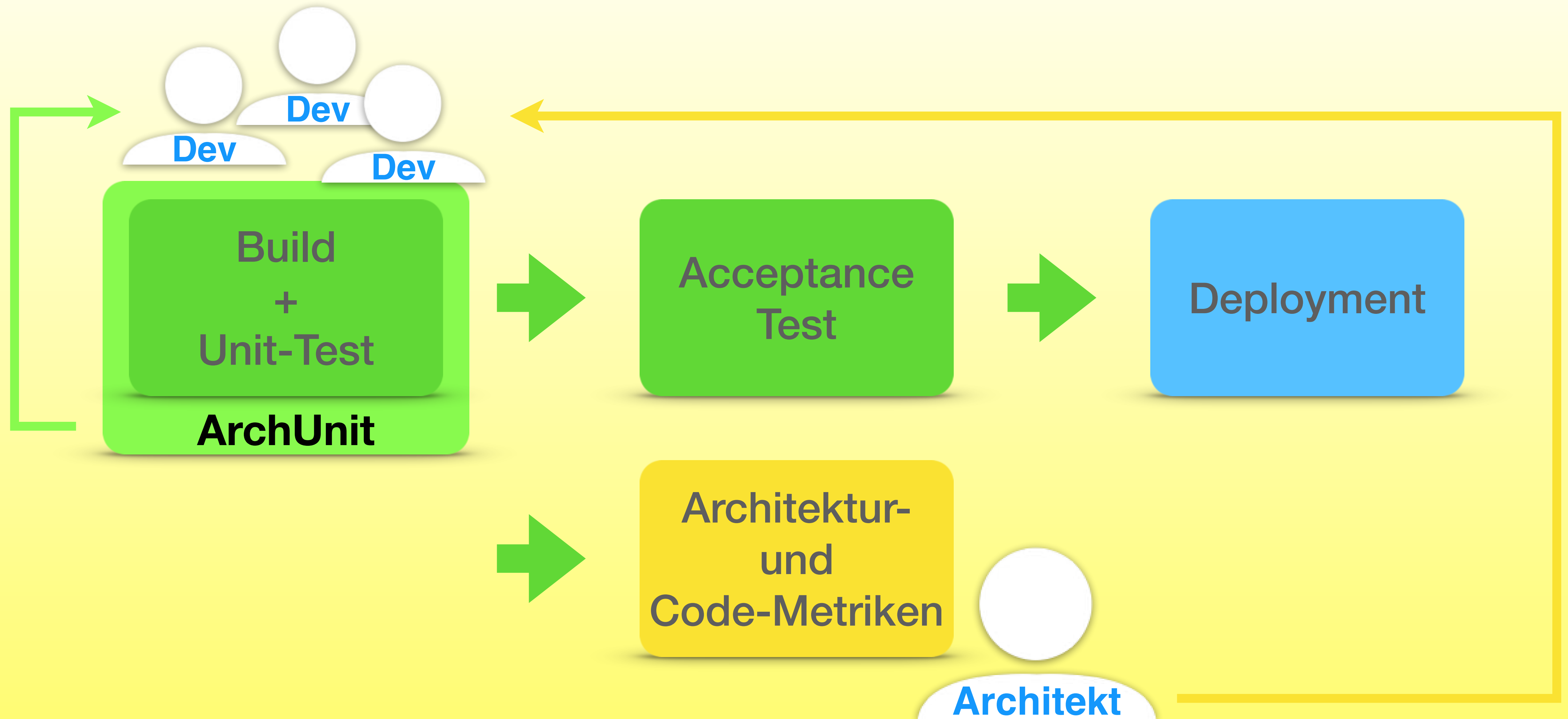
Spezielle Unterstützung für JUnit 4 / 5

Java & Kotlin

Maven- & Gradle-Plugin

Live-Demo

# Warum ArchUnit?



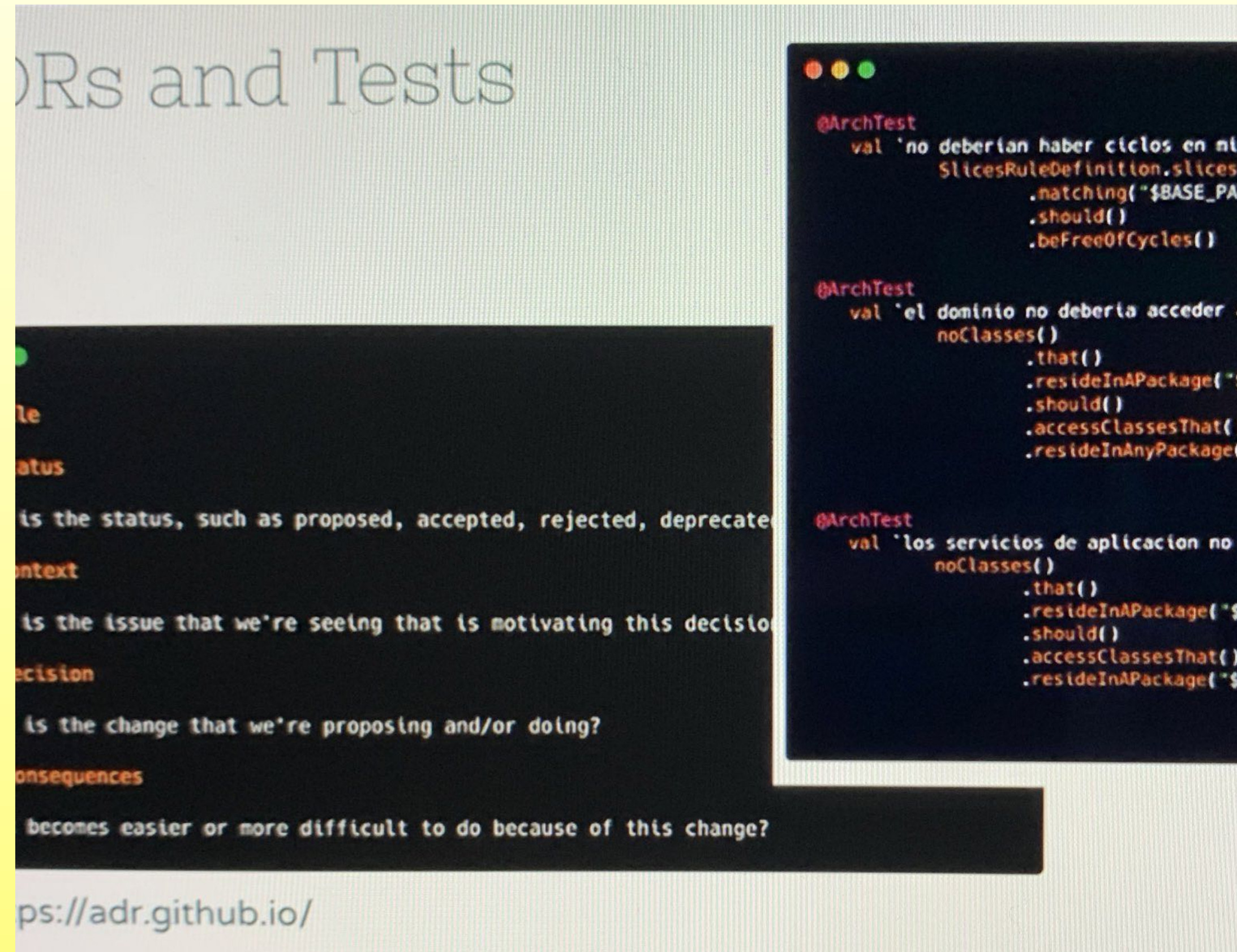
# Architektur-Entscheidungen als Team

adr.github.io

Homepage of the ADR GitHub organization


## Architectural Decision Records

An **Architectural Decision (AD)** is a software design choice that addresses a functional or non-functional requirement that is architecturally significant. An **Architecturally Significant Requirement (ASR)** is a requirement that has a measurable effect on a software system's architecture and quality. An **Architectural Decision Record (ADR)** captures a single AD, such as often done when writing personal notes or meeting minutes; the collection of ADRs created and maintained in a project constitute its *decision log*. All these are within the topic of Architectural Knowledge Management (AKM).



@JavitaLaso beim @ddd\_eu: <https://twitter.com/gtielsch/status/1357716900084674562>

# Alternativen & Ergänzungen

The logo for jQAAssistant features the text "jQAAssistant" in a dark grey font. A green checkmark is superimposed over the "j" and "Q" characters.

The logo for structure101 consists of the text "structure101" in a bold, dark blue, sans-serif font.

The logo for sonarqube features the text "sonarqube" in a dark grey font. To the right of the text are three blue curved lines that resemble a sonar wave.

The logo for checkstyle features the text "checkstyle" in a dark grey font. The letter "e" is stylized as a yellow pencil tip. Below the text is a red wavy line.

The logo for SONARGRAPH features the text "SONARGRAPH" in a dark grey font. The letter "A" is replaced by a red triangle. The entire text is underlined with a black line.

u.v.a.m.



# Modulithen mit Spring Boot

## Moduliths

---

A playground to build technology supporting the development of

build passing

## tl;dr

---

Moduliths is a Spring Boot extension based on ArchUnit to achieve

- *Verify modular structure between individual logical modules*

Prevents cyclic dependencies as well as explicitly defined all to public components in API packages (convention based, c)

- *Bootstrap a subset of the modules of a monolithic Spring P*

<https://github.com/odrotbohm/moduliths>

# jMolecules

## Use Case: Verify and Document Architecture

---

The jMolecules concepts expressed in code can be used to verify rules that stem from the concepts' definitions and generate documentation.

### Available Libraries

- [jQAssistant plugin](#)—to verify rules applying to the different architectural styles, DDD building blocks, CQRS and events. Also creates PlantUML diagrams from the information available in the codebase.
- [ArchUnit rules](#)—allow to verify relationships between DDD building blocks.
- [Moduliths](#)—supports detection of jMolecules components, DDD building blocks and events for module model and documentation purposes (see [blog post](#) for more information).

# https://www.archunit.org

[Getting Started](#)[Motivation](#)[News](#)[User Guide](#)[API](#)[About](#)

Persistence

## Unit test your Java architecture

Start enforcing your architecture within 30 minutes using the test setup you already have.

[Start Now](#)

ArchUnit is a free, simple and extensible library for checking the architecture of your Java code using any plain Java unit test framework. That is, ArchUnit can check dependencies between packages and classes, layers and slices, check for cyclic dependencies and more. It does so by analyzing given Java bytecode, importing all classes into a Java code structure. You can find examples for the current release at [ArchUnit Examples](#) and the sources on [GitHub](#).



*<https://github.com/TNG/ArchUnit-Examples>*

*<https://github.com/thmuch/archunit-demos>*



Schichten

Architektur

Slices

Design

Vertikalen

Konventionen

Fragen?

Monolithen

Module

SCS

Abhängigkeiten

Microservices

Kopplung

Kohäsion



ArchUnit

**Danke!**



Thomas Much

 @thmuch

<https://www.tk.de/it>

