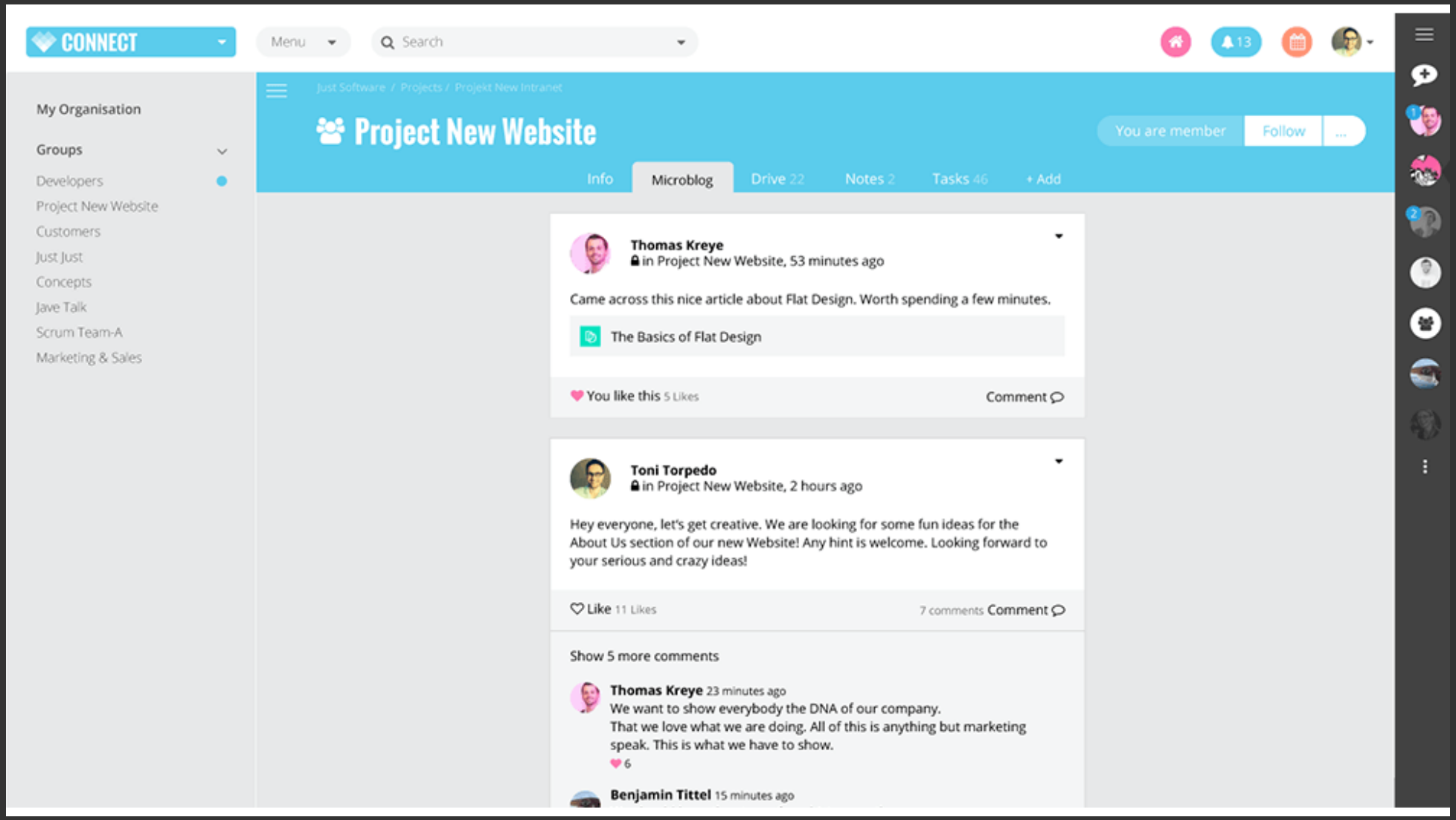
Microservices

Max Jonas Werner / @makkes


LOVE
YOUR
INTRANET

Was ich mache



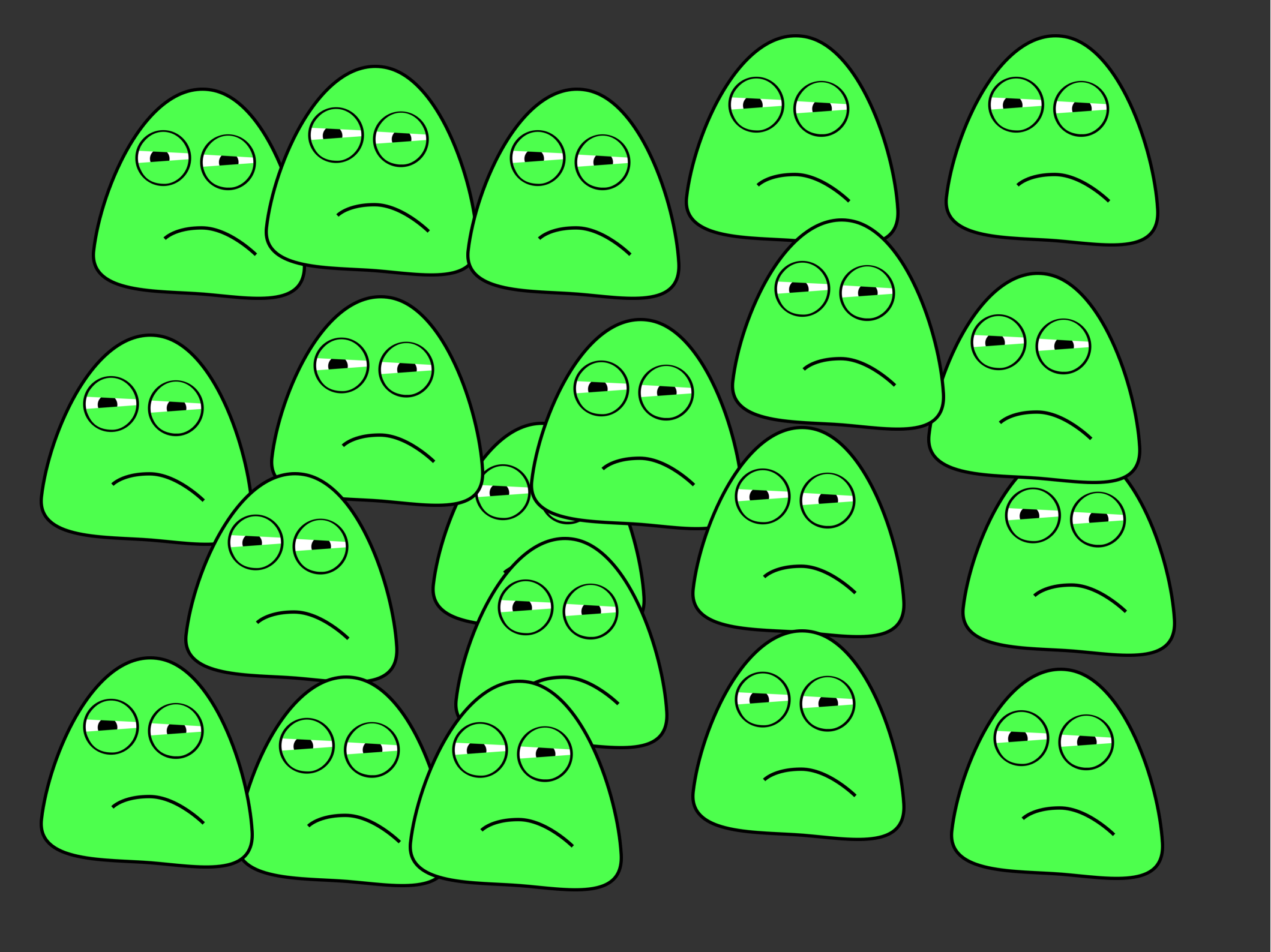
Worum geht's hier?

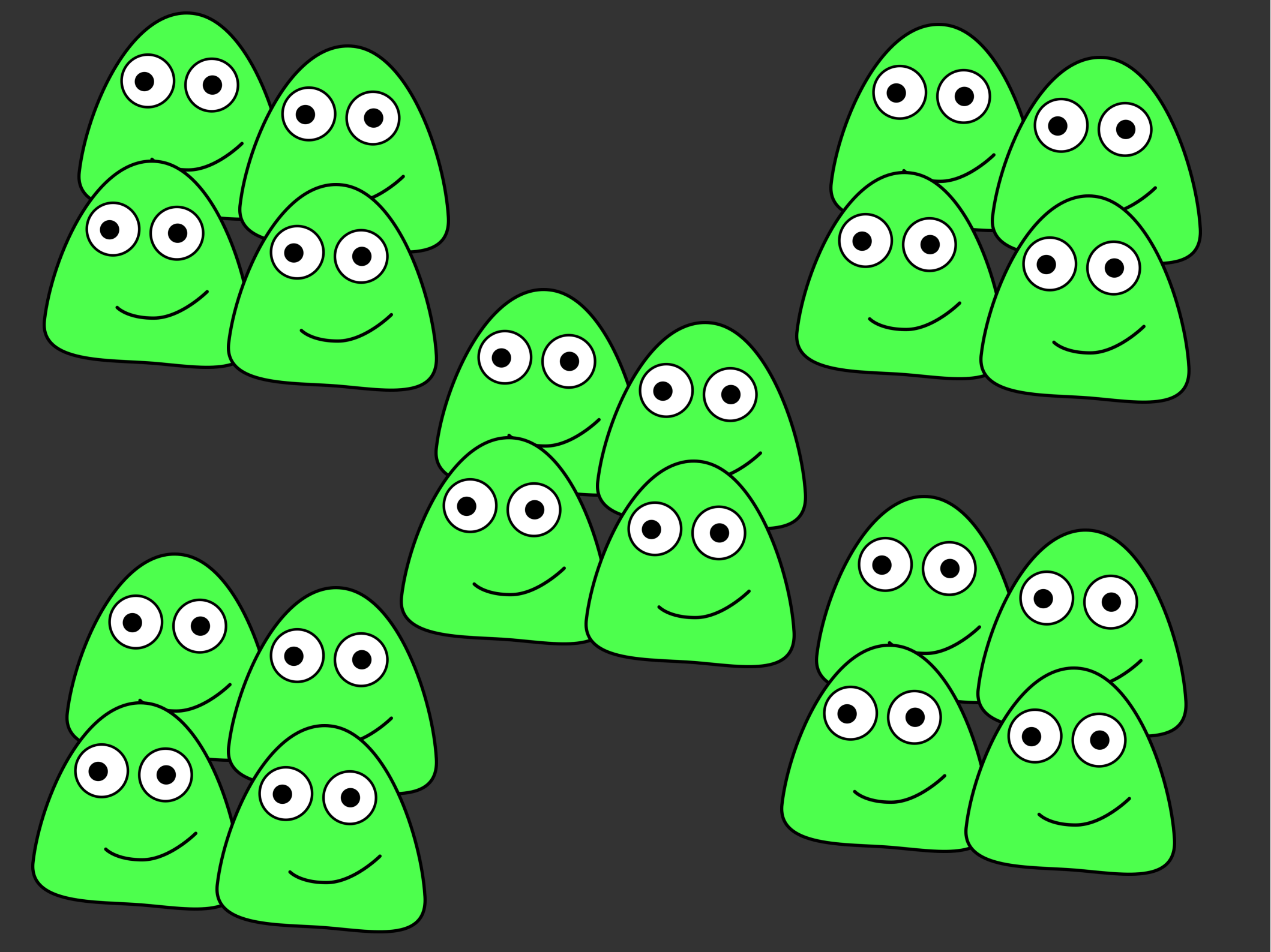
- Software-Entwicklung
- Software-Administration
- Web-Architektur
- Produktentwicklung
- Bingo

Unser Ansatz

- Kurze Entscheidungswege
- Agile Softwareentwicklung
- Nutzung von Open Source
- Vertrauen und Respekt
- Transparente Kommunikation
- Spaß bei der Arbeit


LIEBHERR
AIA



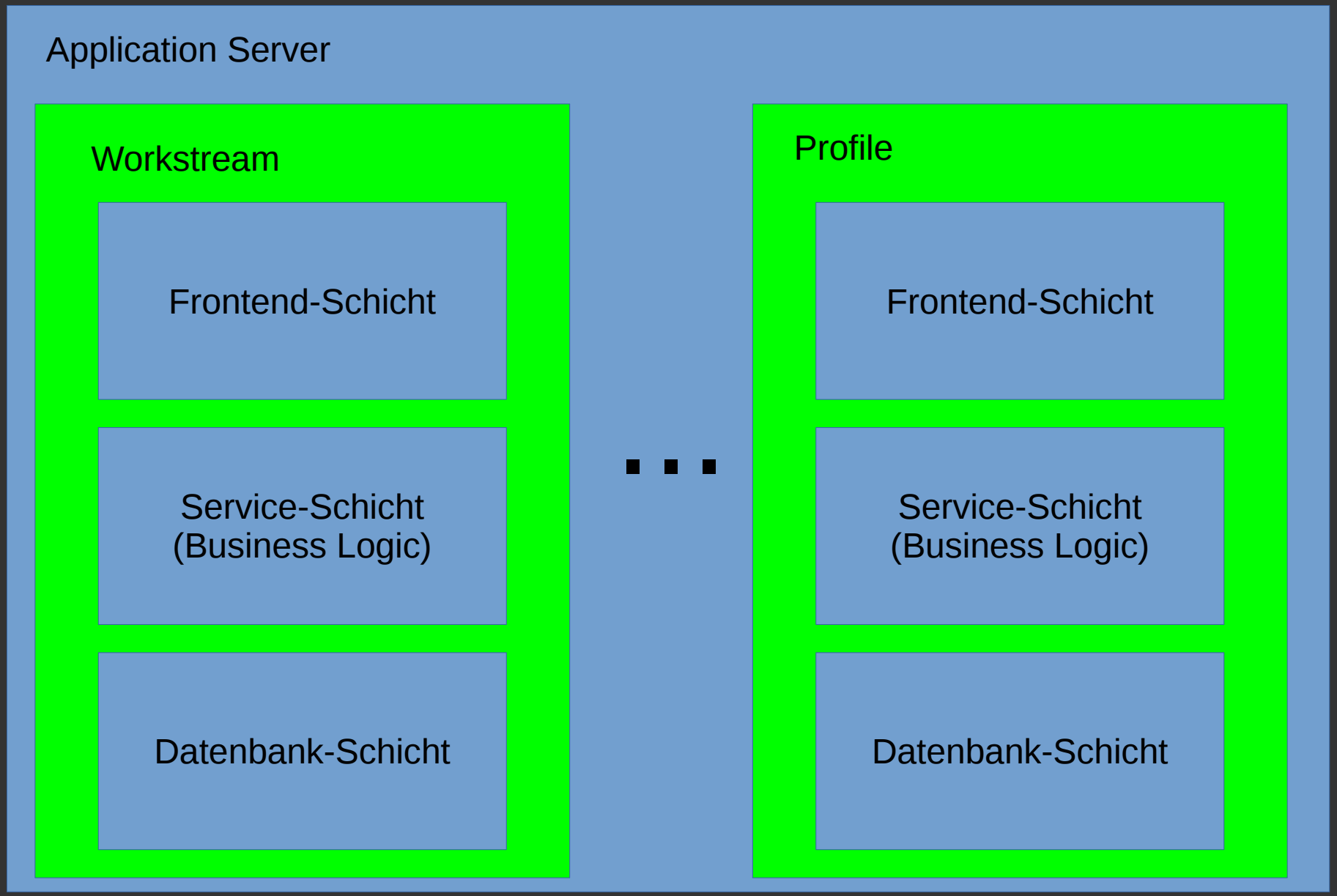


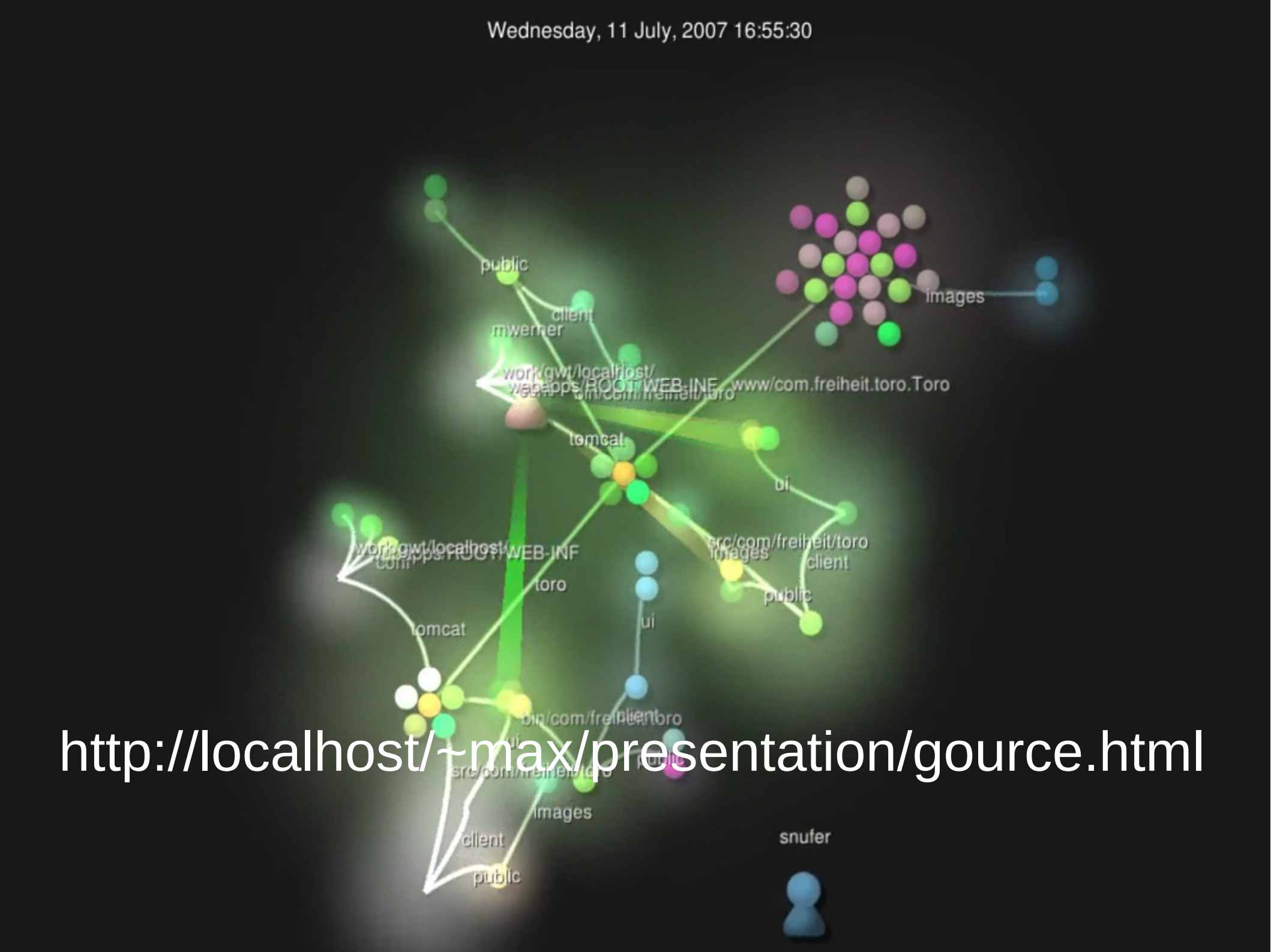

LIEBHERR
AIA

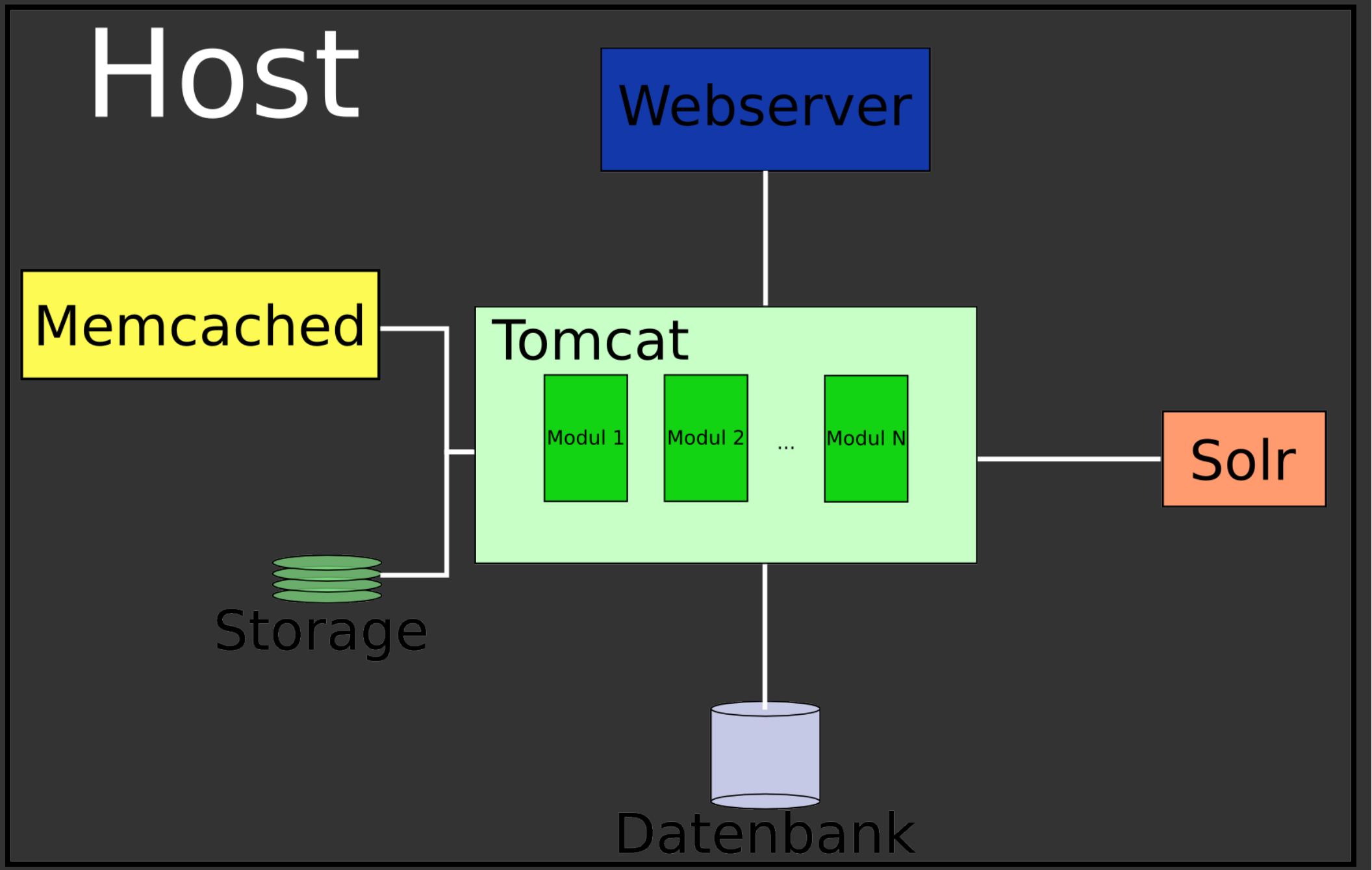
Trotzdem

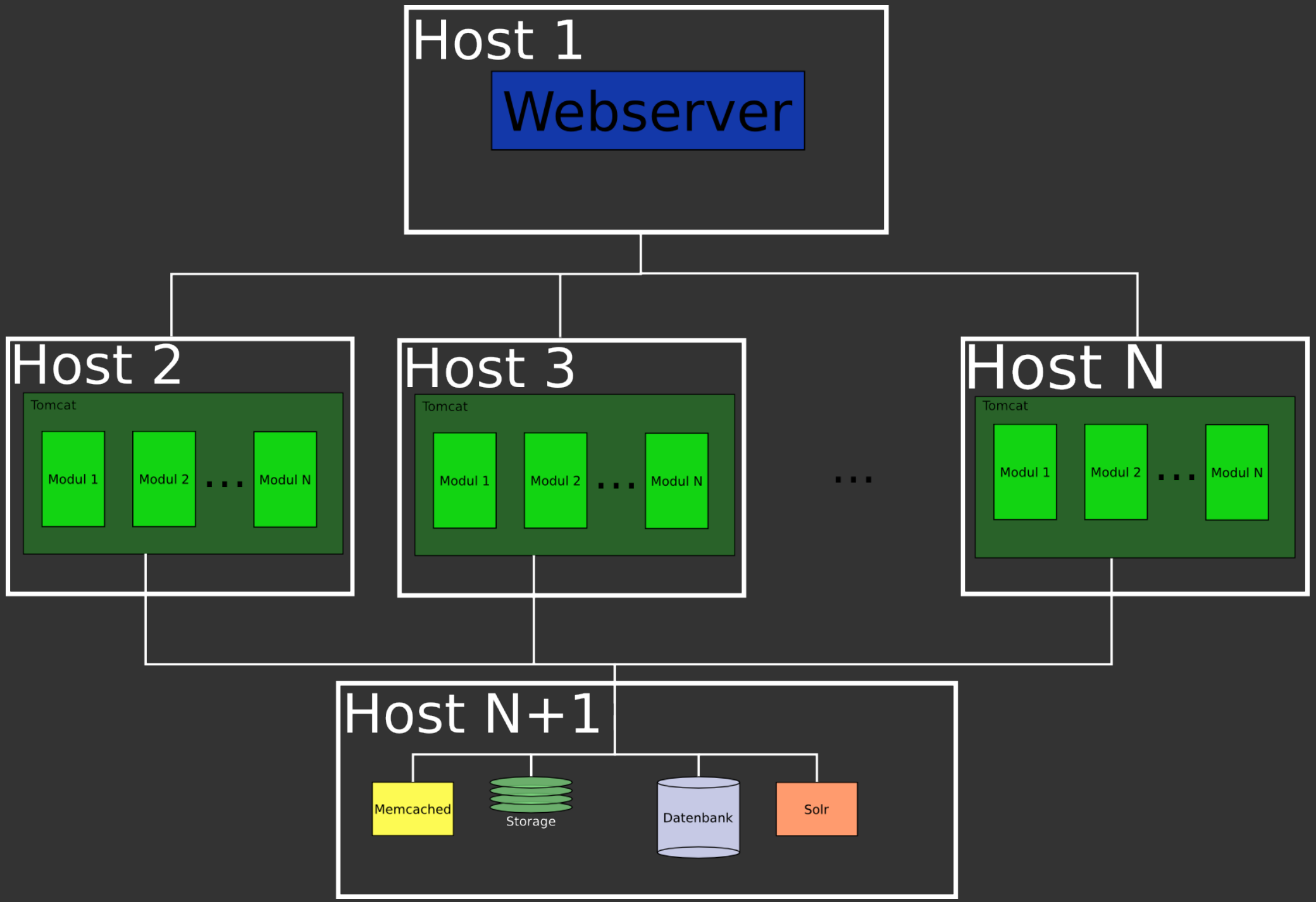
- Keine individuelle Code-Verantwortung
- Teams können wenig selbst entscheiden
- Langsame Entwicklung
- Einsatz von modernen Tools schwierig
- Skalierbarkeit schwierig

Der Monolith




Wednesday, 11 July, 2007 16:55:30
public
client
mwerner
images
tomcat
ui
toro
client
public
tomcat
ui
images
client
public
snufer
http://localhost/~max/presentation/gource.html


Host
Webserver
Memcached
Tomcat
Modul 1
Modul 2
...
Modul N
Solr
Storage
Datenbank


Host 1
Webserver
Host 2
Tomcat
Modul 1
Modul 2
. . .
Modul N
Host 3
Tomcat
Modul 1
Modul 2
. . .
Modul N
. . .
Host N
Tomcat
Modul 1
Modul 2
. . .
Modul N
Host N+1
Memcached
Storage
Datenbank
Solr

Technische Probleme

- Deployments riskant
- Deployments mit hoher Downtime
- Langer Release-Prozess
- Testing aufwändig (App-Context hochfahren etc.)
- Komplexität im Code kaum noch beherrschbar

Technische Probleme

- Continuous Delivery fast unmöglich
- Wie führen wir neue Technologien ein?
  - Programmiersprachen
  - Frameworks
  - …
- Was passiert, wenn wir wachsen?
- Skalierbarkeit der Organisation schwierig

Conway's Law

“Organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations.” – Melvin E. Conway

Microservices

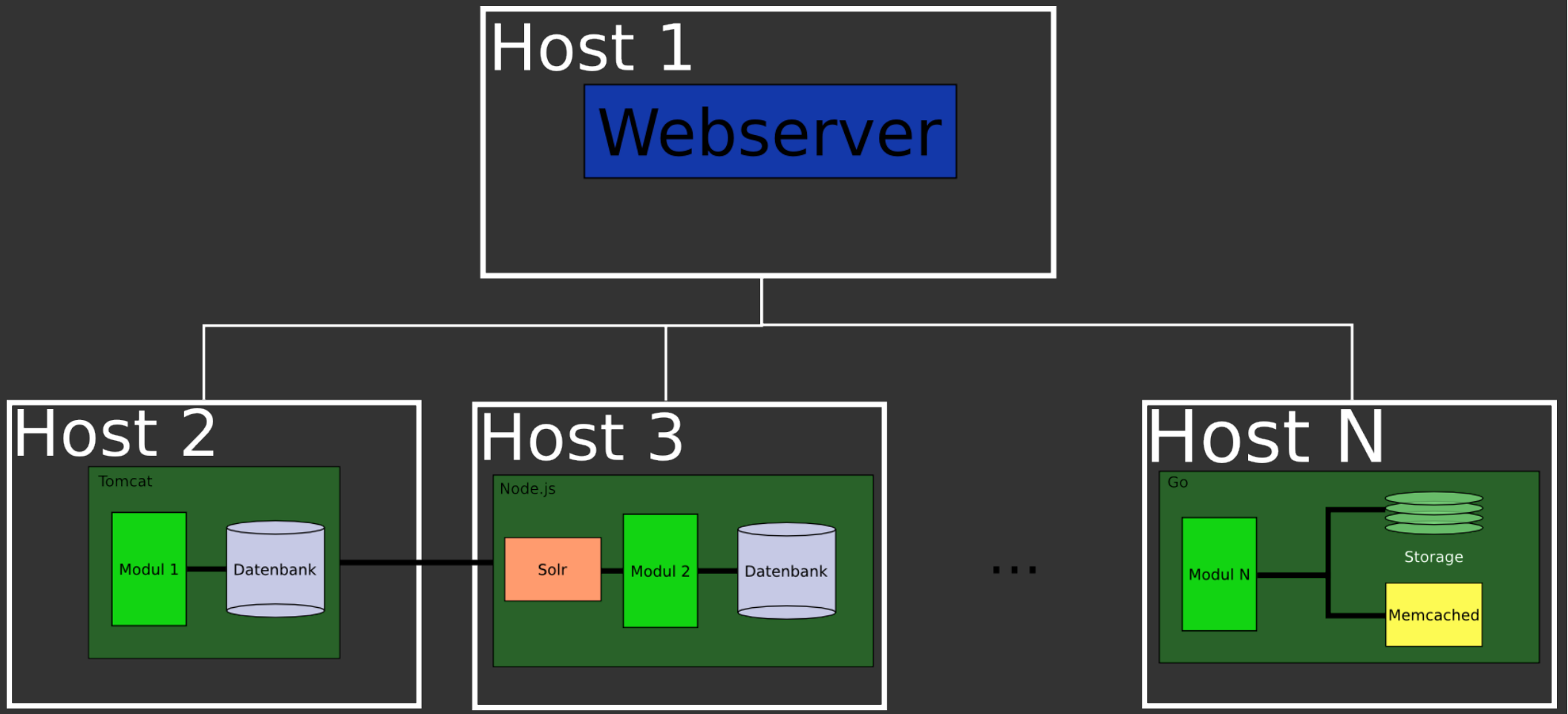
- Was, wenn jeder Service in sich geschlossen wäre?
- Services werden lose gekoppelt
- Kommunikation über definierte APIs
- Vom Monolithen zum verteilten System

Das hatten wir doch schon mal?

- SOA
- CORBA
- Warum hat das nicht funktioniert?

Definition

"In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies." — Martin Fowler


Host 1
Webserver
Host 2
Tomcat
Modul 1
Datenbank
Host 3
Node.js
Solr
Modul 2
Datenbank
...
Host N
Go
Modul N
Storage
Memcached

Vorteile

Unterschiedliche Technologien

Vorteile

Ausfallsicherheit

Vorteile

Individuelle Skalierung (Cloud)

Vorteile

Individuelles Deployment

Vorteile

Neue Features

Risiken

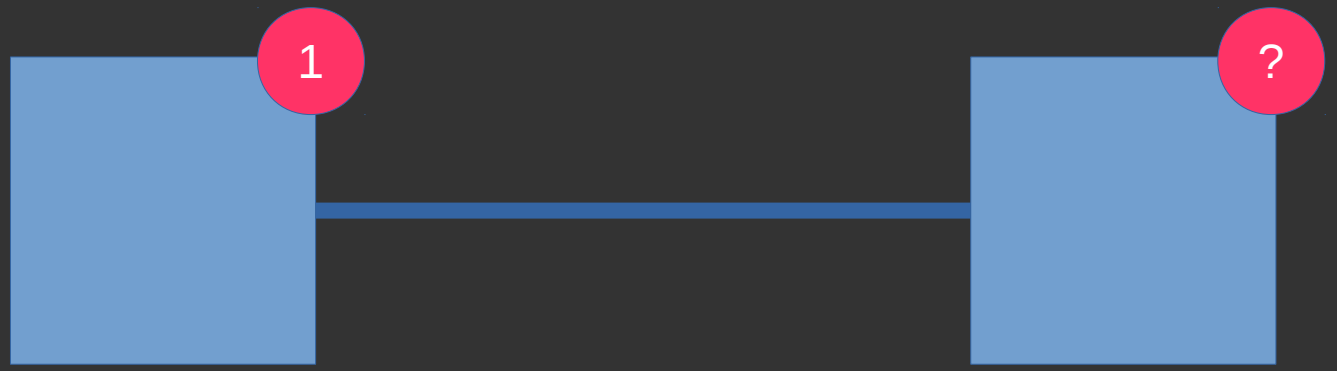
Kommunikation ist langsam

Risiken

Kommunikation ist fehleranfällig

Risiken

Datenhaltung inkonsistent



Risiken

Betriebskomplexität

- Deployment
- Monitoring
- Debugging
- Orchestrierung

Prinzipien

- Redundante Datenhaltung
- Keine synchronen Requests während einer Interaktion
- Möglichst asynchrone Kommunikation
- Lose Kopplung
- Stabile APIs

Technologien

- Inter-server-Kommunikation: G-RPC, Thrift, HTTP/REST
- Datenserialisierung: JSON, XML, ProtoBuf, Hypermedia
- Deployment: Docker

Management

- Autonome Teams
- Vertrauen
- 100% Verantwortung (“you build it, you run it”)
- DevOps-Mentalität

Wann Microservices?

- “There is no silver bullet”
- Start mit Monolith (Time to Market)
- Prepare for Failure
- Lerne auf dem Weg

max@just.social