SEBA Master - Web Application Engineering

“Technology Revolution @BAUHAUS“

Session hosts:

Matthias Riepl

ITNanny@BAUHAUS

Seyit Hannemann

Tech Lead@BAUHAUS

BAUHAUS in a nutshell

- First BAUHAUS opened in Mannheim in 1960
- By now 159 BAUHAUS stores in Germany
- And overall 270 stores in 19 european countries
- Product range with more than 120000 products
- And still growing..

BAHAG is the e-business division for BAUHAUS.
We are committed to digitalization as a means of creating a digital BAUHAUS of the future.

Where we stand and where we wanna go

1990's

SPAGHETTI-ORIENTED
ARCHITECTURE
(aka Copy & Paste)

Present and future

PIZZA-ORIENTED
ARCHITECTURE
(aka Cloud, µ, API)

As-Is:

- No real internal dev capa
- Special tasks only
- Customization mostly done external
- Best-of-breed: COTS

To-Be:

- Homegrown (own it)
- Cloud native, serverless
- Microservice-cut
- Platform-based (abstracted)

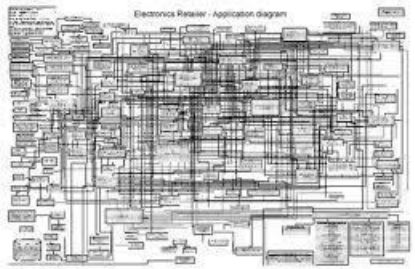
→ For everything which is differentiating

BAUHAUS

The general migration approach is adoption before migration

As-Is

Heterogenous app landscape
- redundancies
- complex
- divers technologies
- …



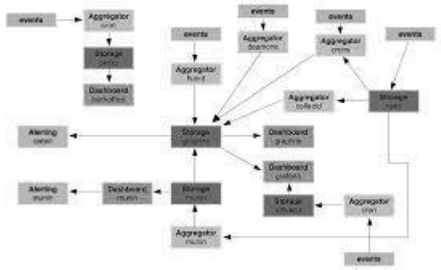
Two transformation options

1
Adoption
(refactor & repurchase)
"build new capabilities on cloud"

2
Migration
(rehost & replatform)
"shift existing app to cloud"

To-Be

public-cloud based modular landscape
- Standard tooling
- clear architecture and quality principles
- …



We strongly prefer „Adoption", although it is the most complex option, outcome and clarity are superior.

BAUHAUS

But is it only about technology? A holistic approach to realize Omnichannel is needed

1

Strategy

A shared purpose and vision embodies across the organization

2

Structure

A network of empowered teams

3

Process

Rapid decision and learning cycles

4

People

Dynamic people model that ignites passion

5

Technology

Next-generation enabling technology

https://www.mckinsey.com/business-functions/organization/our-insights/enterprise-agility-buzz-or-business-impact

BAUHAUS

What is a „Product“?

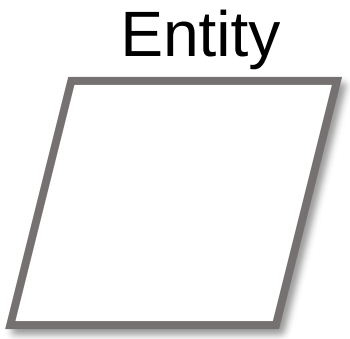
In a nutshell

Product = Customer x Business x Technology

https://svpg.com/

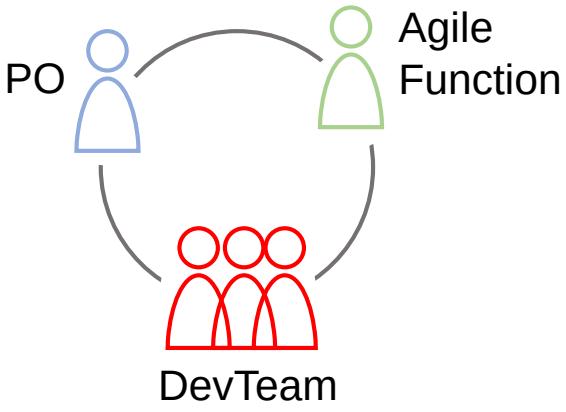
BAUHAUS

Product Details: Organisation



Organisational Unit

- Cost Center ID
- Org Unit ID

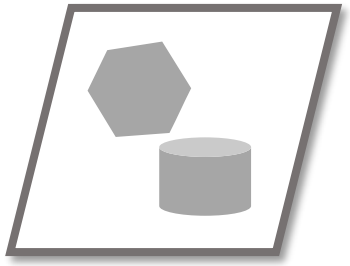


Agile Software DevTeam

- PO = Value
- AF = Flow
- Dev = Quality

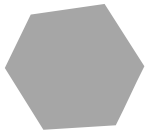
Product Details: EA view

Entity

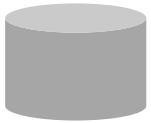


Each product owns specific capabilities & objects.
There is a unique relation.

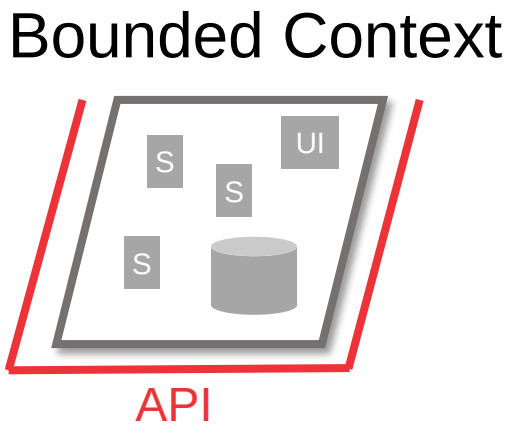
› Owns business capabilities



› Owns business objects



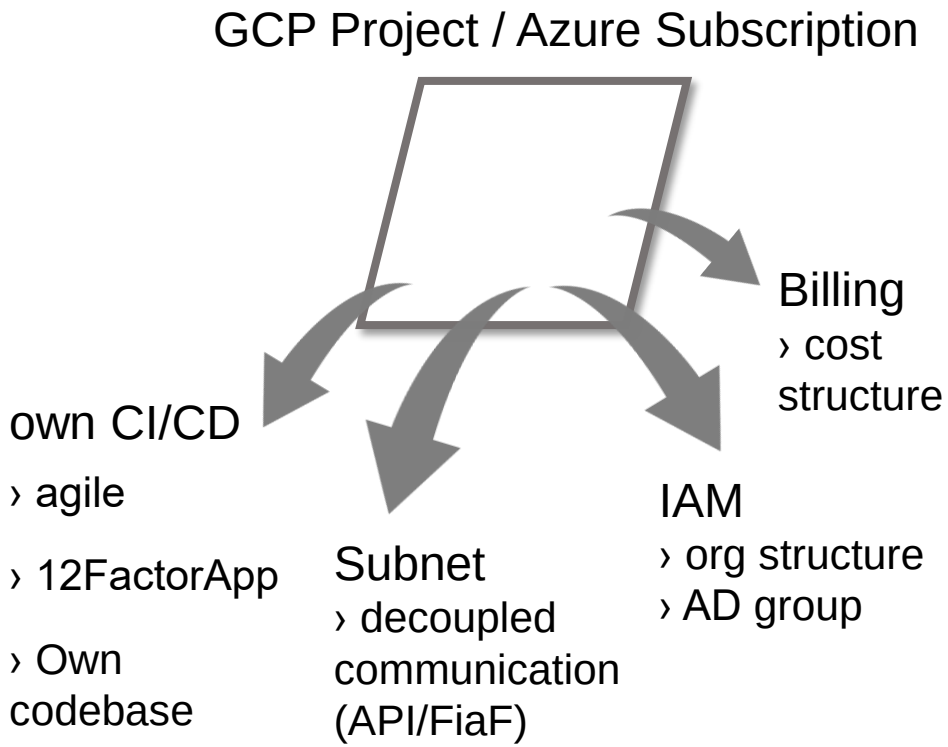
Product Details: Application view



Following DDD-principles & common microservice approaches.

- It`s a bounded context
- Decoupled via an API
- Clearly described by consumer driven contracts

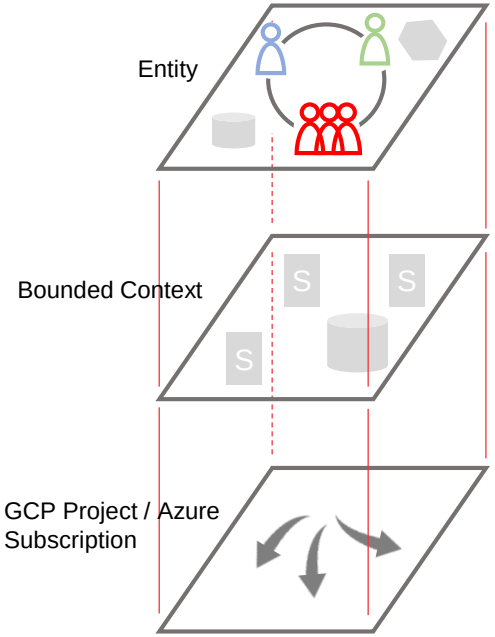
Product Details: Technology view



A ‘unit’ in the cloud

- Ownership area
- Self-enablement
- No dependencies
- Internal view decoupled

Product Details Summary: It‘s vertical!



Org & EA

MS & API

Cloud & Technology

Minimize dependencies ⇨ Aligned Autonomy

Now we zoom out: Macroview

⇨ Macro = Enterprise view or omni-channel approach (adoptability)

Touch Point	SocialX	Price Hubs	Shop	App	…
Logic xc	Search	Checkout	Pay	…	
Data	Product	Price	Customer	…	
Tech / Platforms	Monitor	Compute	Integrate	…	

Key principles

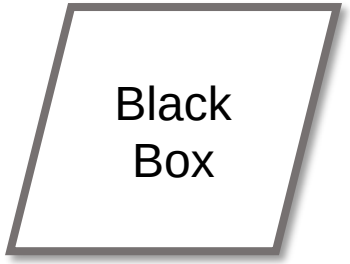
- Segregation of duties
- Clear Ownership & responsibilities
- Decoupling via API
- Pull from customer

Simple but clear frame

Follow this link to learn more about Macro and Micro Architectures, and ISA

BAUHAUS

Now we zoom in: Microview



You as a product have clear borders & clearly defined functionalities & objectives, which are described with customer driven contracts.

From outside it`s a black box.

But inside is FREEDOM

- Internal structure
- Tech
- Architecture
- Roadmap

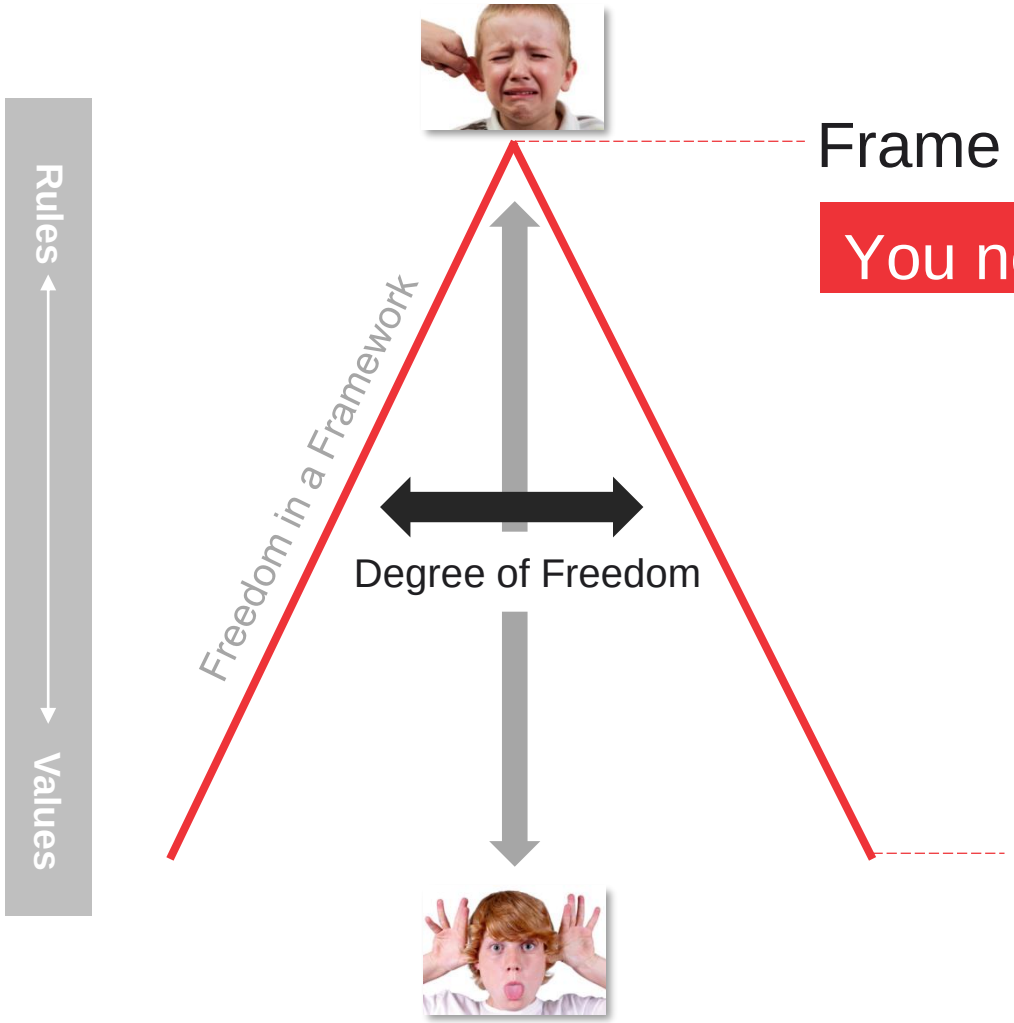
OKRs

Vegas-rule: What happens in a product, stays in a product!

Follow this link to learn more about Macro and Micro Architectures, and ISA

BAUHAUS

Combining Macro & Micro: FiaF



Frame (Macro)

You need to align on Enterprise level

You are free to decide on Product level

Freedom (Micro)

Full freedom on product level?

There are three drivers which ‚direct' your freedom

1 OKR

- Clear focus
- Value-oriented
- Outcome-driven
- Aligned

2 Principles

- Operational Excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization
- Scalability per design

3 Technology

- Adoptability
- Interchangeability
- Knowledge monopoly
- InnerSource or re-usability
- Segregated

Follow this link to learn more about cloud-native programing standards

BAUHAUS

But is this enough?

But is this enough?

Now: We need to detail the TechStack, but first let`s analyze where we stand:

1. Please visit Bauhaus.info!

What do you think are our biggest pain points?

But is this enough?

Now: We need to detail the TechStack, but first let`s analyze where we stand:

1. Please visit Bauhaus.info!
2. Please check our SEO performance!

What do you think are our biggest pain points?

BAUHAUS

But is this enough?

Now: We need to detail the TechStack, but first let`s analyze where we stand:

1. Please visit Bauhaus.info!
2. Please check our SEO performance!
3. Please check our load performance!

What do you think are our biggest pain points?

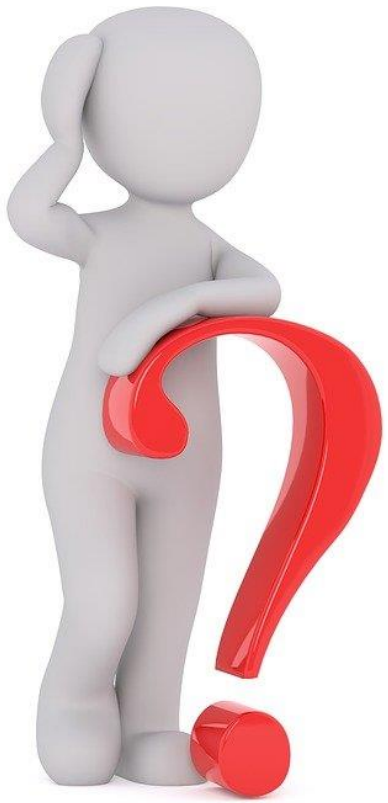
BAUHAUS

Pain@Shop

Let’s collect!

List of pain points:

- No customer reviews
- No availability
- Slow
- Poor SEO
- Product details plain (content)
- Personalization zero
- No mobile xp
- …



Future FE-frame needs to fulfill:

What we want to have:

- User-specific
- Fast
- Low cost
- High dev-speed
- Multi-device support
- SEO optimized

Which stack to choose?:

?

BAUHAUS

Now you know the pain, how can we tackle that?

Vote for:

React

Ember

Angular

Vue

Svelte

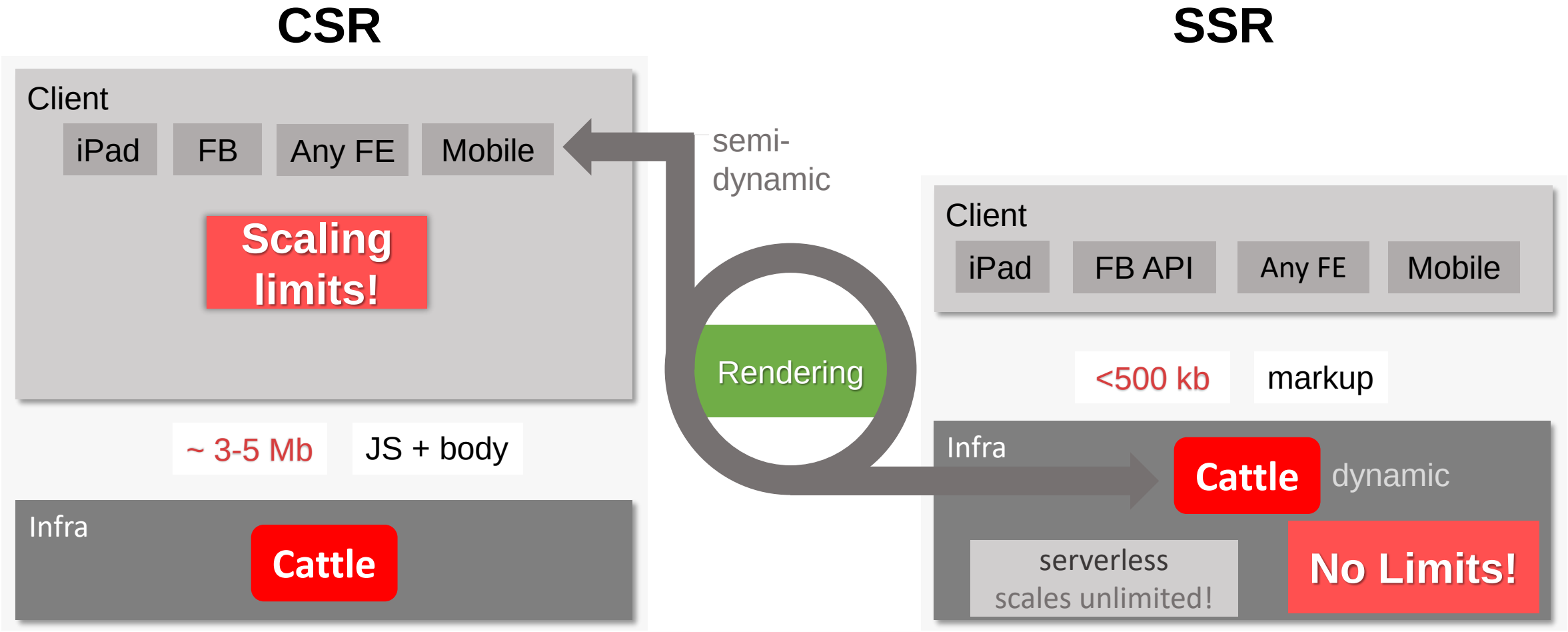
Something different

Please vote on mentimeter!

https://www.mentimeter.com/s/23c6e6430e4d4ced190b79dfd3b1b14f/abe69ed00b1c/edit

BAUHAUS

Our Thoughts: Client time is over – Server rule again



https://gist.github.com/Restuta/cda69e50a853aa64912d

Future FE-frame needs to fulfill:

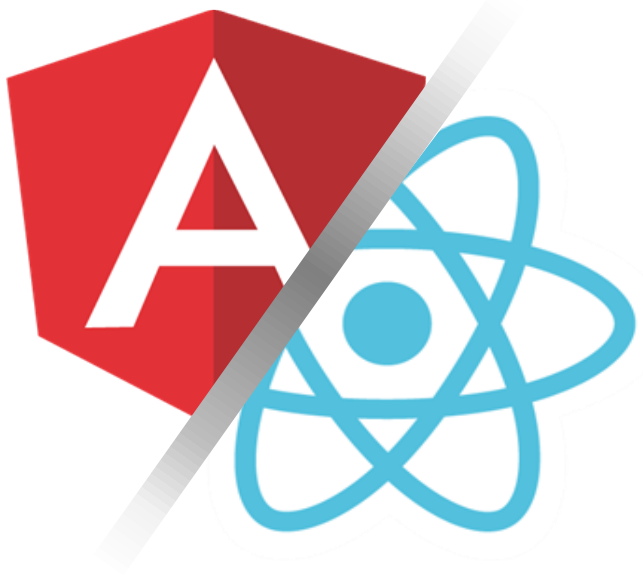
What we need:

	SSR provides:
User-specific	Highly personalized (flexible content, dynamic)
Fast	Fast pages, low load times
Low cost	Less traffic + higher scale
High dev-speed	Low Cycle & Lead time
Multi-device support	SSR – flat, fast, flexible (fff)
SEO optimized	SST plus higher SEO potential (SSR)

BAUHAUS

Finally, go with React for Shop-E-Frame

Angular (SSR flansched)	React (native SSR)
Big packages (3-5 Mb) - not as fast - possibly crucial for mobile	Small packages (<500 kb) - fast - connection independent (GPRS)
Complex FE‘s easier to build	Better UX on all devices and light weight (mobile focus)
Rendering on client neccessary - takes time, (slow) JS	Already rendered - fast, speed ++
SSR not native - overhead, boiler	Client-side code in backend - easy, 1:1
Predefined components (Ang) - fast ramp up	Plain, total flex - needs more time to ramp up



https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/

BAUHAUS

Compare several csr-ssr approaches

Server ←- → Browser

	Server Rendering	“Static SSR”	SSR with (Re)hydration	CSR with Prerendering	Full CSR
Overview:	An application where input is navigation requests and the output is HTML in response to them.	Built as a Single Page App, but all pages prerendered to static HTML as a build step, and the JS is **removed**.	Built as a Single Page App. The server prerenders pages, but the full app is also booted on the client.	A Single Page App, where the initial shell/skeleton is prerendered to static HTML at build time.	A Single Page App. All logic, rendering and booting is done on the client. HTML is essentially just script & style tags.
Authoring:	Entirely server-side (request-response, HTML)	Built as if client-side (components, DOM*, fetch)	Built as client-side	Client-side	Client-side
Rendering:	Dynamic HTML	Static HTML	Dynamic HTML **and** JS/DOM	Partial static HTML, then JS/DOM	Entirely JS/DOM
Server role:	Controls all aspects. (thin client)	Delivers static HTML	Renders pages (navigation requests)	Delivers static HTML	Delivers static HTML
Pros:	👍 TTI = FCP 👍 Fully streaming	👍 Fast TTFB 👍 TTI = FCP 👍 Fully streaming	👍 Flexible	👍 Flexible 👍 Fast TTFB	👍 Flexible 👍 Fast TTFB
Cons:	👎 Slow TTFB 👎 Inflexible	👎 Inflexible 👎 Leads to hydration	👎 Slow TTFB 👎 TTI >>> FCP 👎 Usually buffered	👎 TTI > FCP 👎 Limited streaming	👎 TTI >>> FCP 👎 No streaming
Scales via:	Infra size / cost	build/deploy size	Infra size & JS size	JS size	JS size
Examples:	Gmail HTML, Hacker News	Docusaurus, Netflix*	Next.js, Razzle, etc	Gatsby, Vuepress, etc	Most apps

https://developers.google.com/web/updates/2019/02/rendering-on-the-web

OK, React – but where should it run?

	VMs	Containers	Serverless Computing
+	› Extensive management › Well understood	› Flexible › Easily mixed and matched	› Lightweight › Portable
-	› Tied to OS › Less portable	› Tied Difficult to manage › Generates large volumes of small pieces of code	› Vendor lock-in* › Not widely used

Client Serverless Revolution

Client-serverless has roots in the old-guard, three-tier application architectures that sprung up around PCs and local area networks that connected a client-side GUI to a back-end SQL database. But this new paradigm is much better suited to **21st century multicloud computing platforms**. This is because client-serverless:

- Delivers composable functions **at low latency via a consistent, secure, web-native API** that can be called from any client application and on a pay-as-you-go basis.
- Enables applications to be **easily served, composed, and consumed on demand** from every piece of computing infrastructure anywhere.
- Allows developers to **deploy functions quickly and scalably** across cloud-to-edge environments.
- Ensures that **application performance won't degrade** even as the underlying business logic is distributed far and wide.
- **Abstracts away the physical locations** and operating platforms from which the back-end application logic is being served.
- **Eliminates the need for programmers to write the logic** that manages containers, virtual machines, and other back-end runtime engines to which execution of application logic will be dynamically allocated.
- **Boosts the density, efficiency, and capacity utilization of CPU, memory, storage,** and other hardware utilization on the back-end cloud platforms.

https://www.infoworld.com/article/3604674/welcome-to-the-client-serverless-revolution.html

Tech stack must match agile dev principles

We are

- Serverless
- NoOps
- Event-driven

And these are the technologies:

Node.js
React
JavaScript
Dart
LINUX
Phyton
GCP
Azure
Kubernetes
Datadog
Google BigQuery
GitHub
REST API
TypeScript
git
redis
Cloud RUN
JIRA
JAVA
Flutter

BAUHAUS

It`s not only about tech, also the people using it

- Don‘t reinvent the wheel!
- KISS & YAGNI!
- Eat your own dogfood!
- Fail fast, fix forward!
- You build it, you run it, you fix it!

and always ask the customer

BAUHAUS

To increase the level of enterprise agility, companies face implementation choices across five dimensions.

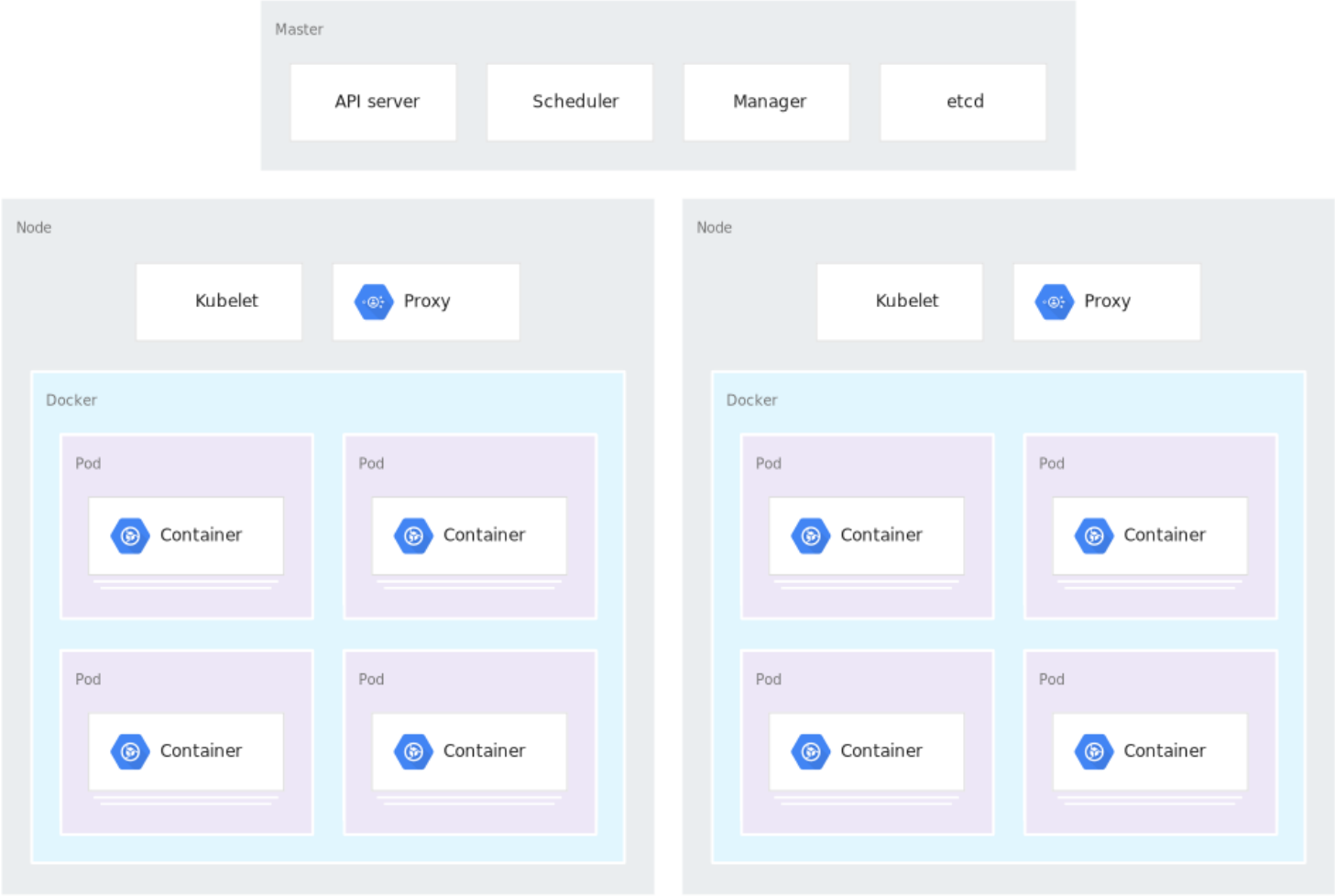
1	2	3	4	5
Strategy	**Structure**	**Process**	**People**	**Technology**
A shared purpose and vision embodies across the organization	A network of empowered teams	Rapid decision and learning cycles	Dynamic people model that ignites passion	Next-generation enabling technology
OKR	**Product Organisation**	**Scrum, Kanban, …**	**Agile values, FiaF, attitude**	**Cloud, API, µServices**

BAUHAUS

But enough paperwork, demotime!

BAUHAUS

But enough paperwork, demotime!

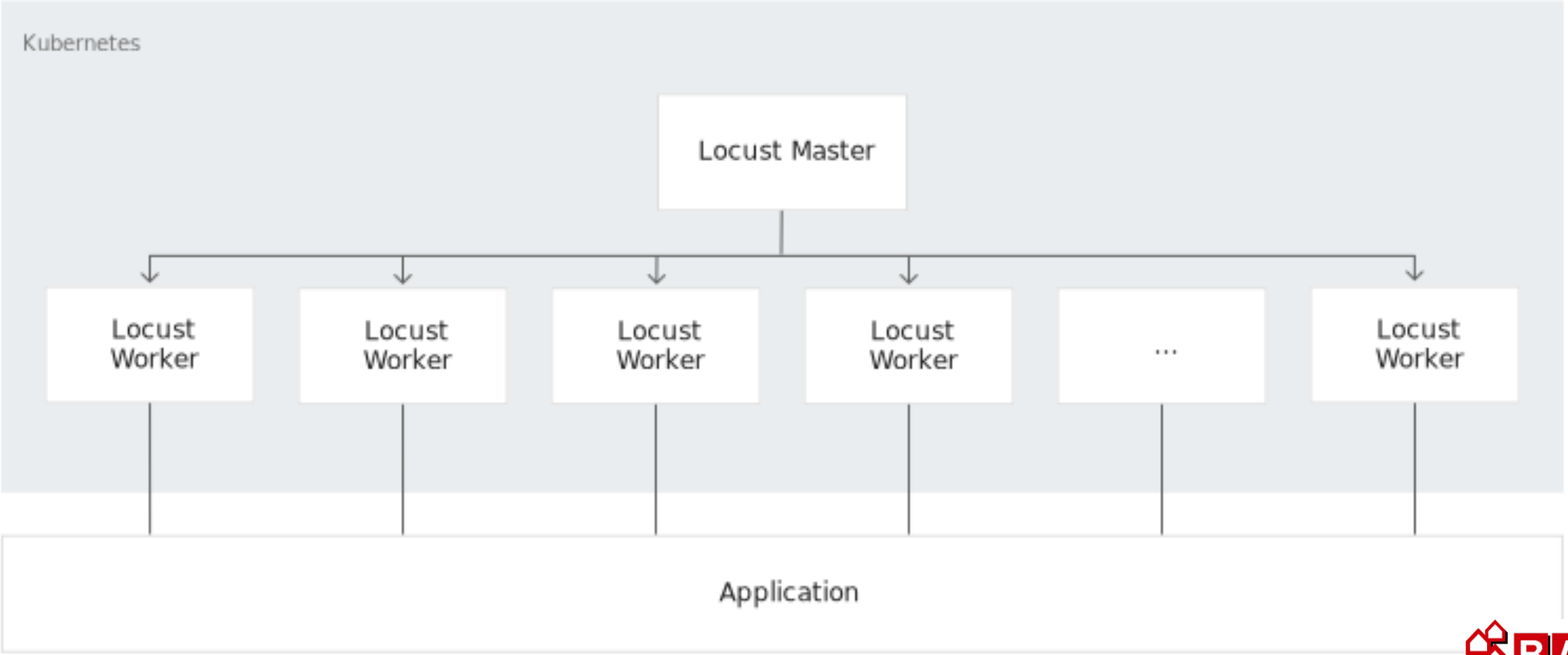


BAUHAUS

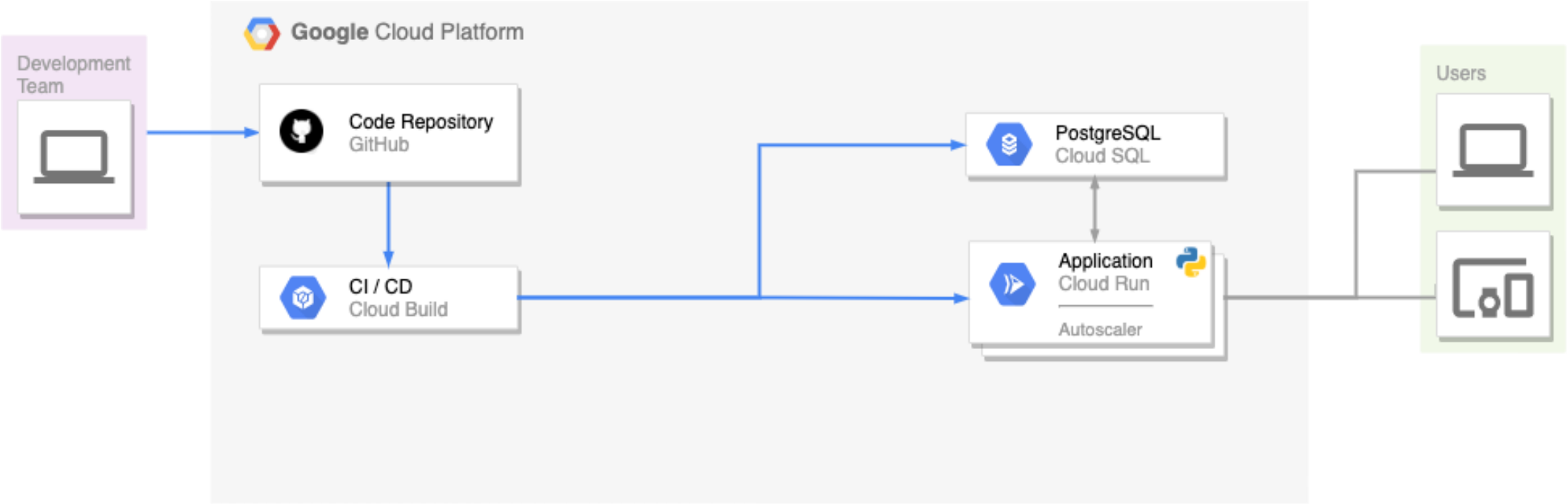
But enough paperwork, demotime!

Locust

- Distributed & scalable
- Define user behaviour in code
- Proven & battle tested



But enough paperwork, demotime!



BAUHAUS

Was built by Trainees in 2 days …

- Overall 18 Months
- Join three product teams (each 6 months)
- Continuous mentorship
- 85 days for trainings (20%)
  - Coding training (cloud-native, serverless, JAMStack,…)
  - Agile training
  - Business training

Click here for more details

APPLY NOW!

BAUHAUS

THANKS!

Q&A?

https://www.menti.com/6fb5gn1k9e

Operationalizing strategy: OKR!

Scrum/Kanban | **OKR**

Task ⟷ Output ⟷ Outcome ⟷ Impact