Moderne Software-architektur-dokumentation

Falk Sippach
embarc

Ralf D. Müller
DB Systel GmbH

Moderne Softwarearchitekturdokumentation

Architekturdokumentation wird sehr oft stiefmütterlich behandelt. Dabei unterstützt das Dokumentieren bei der Entwurfsarbeit, schafft Transparenz bzw. Leitplanken für die Umsetzung und Wartung der Softwarearchitektur. Einerseits ist es aber nicht einfach, die wichtigen Informationen aus dem Entwurf der Softwarearchitektur strukturiert und leicht verständlich festzuhalten. Andererseits enden die meisten Versuche auf der Suche nach einer praktikablen Handhabung zur Erstellung und Pflege in der WYSIWYG-Hölle einer Textverarbeitung oder im tiefen Schlund eines Wikis. Dieses Tutorial zeigt aufbauend auf leichtgewichtigen Tools und Textformaten die Erstellung einer möglichst redundanzfreien Dokumentation, die für verschiedene Zielgruppen optimiert in ansprechenden Formaten ausgeliefert werden kann.

Anhand von vielen praktischen Übungen geht es um Begriffe wie Continuous Documentation und Documentation as Code. Das Ziel ist die moderne, effektive und pragmatische Dokumentation der Softwarearchitektur. Wir bauen auf bewährte Methoden, Formate und Tools wie AsciiDoc/Markdown, PlantUML, docToolchain, Maven/Gradle und viele weitere. Im Detail kümmern wir uns um die Automatisierung des Dokumentations-Build-Prozesses, das Generieren von Inhalten aus dem Modell, Datenbankschema oder Sourcecode, die strukturierte Ablage inklusive Versionier- und Historisierbarkeit und die Verwendung bzw. Erstellung von aussagekräftigen und einfach wartbaren Grafiken.

Agile Entwicklungsteams können so die Dokumentationsarbeit in ihre täglichen Aufgaben integrieren und jederzeit aktuelle, umfassende und gut strukturierte Ergebnisse ausliefern. Zudem lässt sich die Erstellung der Dokumentation in den Reviewprozess integrieren und so stetig verbessern und weiterentwickeln.

08:30 - 09:20

Documentation as Code

docToolchain einrichten

10 min

Pause!

09:30 - 10:20

Architektur dokumentieren

Übungen

10 min

10:30 - 11:20

Praktische Umsetzung

Coding

10 min

Fragen?

11:30 - 12:20

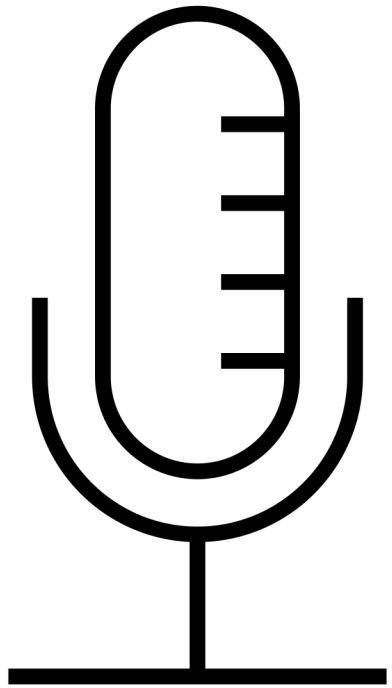
Was gibt es noch?

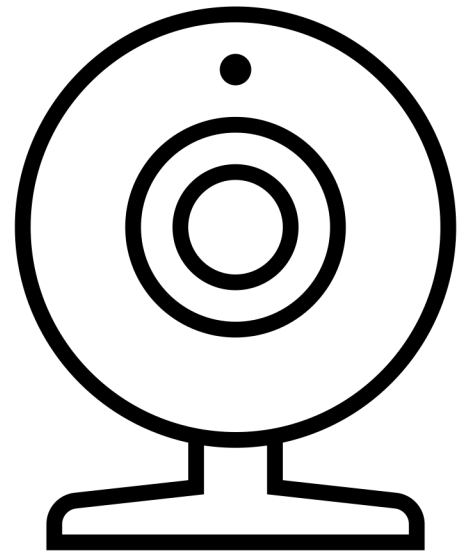
Demos

10 min Fragen

Übungszeit

Theorie





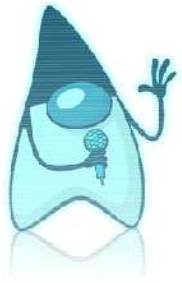
Falk Sippach

- Softwarearchitekt, Berater, Trainer bei embarc
- früher bei Orientation in Objects (OIO), Trivadis

fs@embarc.de

@sippsack

➔ xing.to/fsi



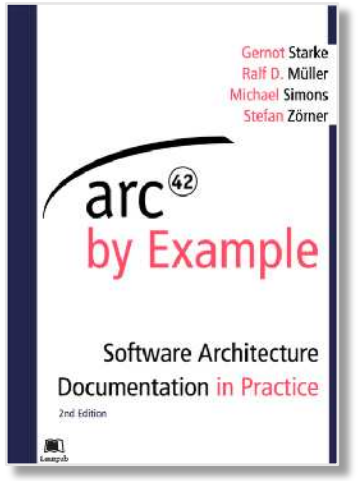
Ralf Müller

- was mit Dokumentation, Security und Architektur by der DB Systel
- Maintainer von docToolchain, Contributor arc42
- co-Autor

ralf.d.mueller@deutschebahn.com

@ralfdmueller

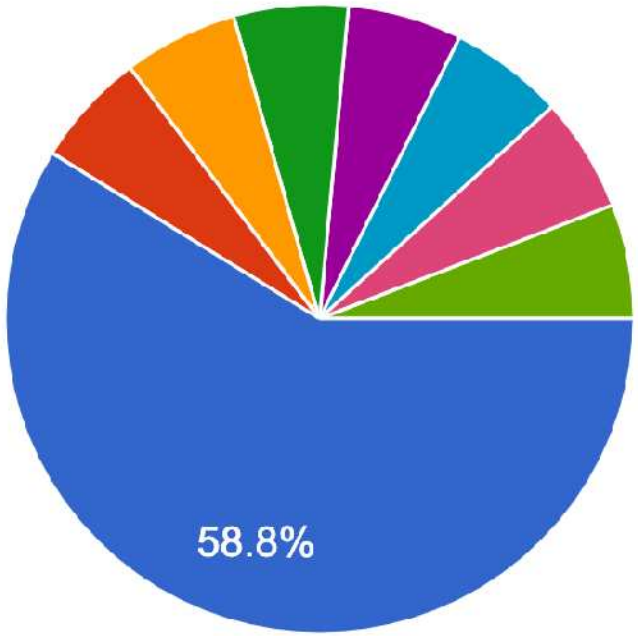
➔ xing.to/rdmuller



Wer seid Ihr?

Was bezeichnet Deine Rolle am Besten?

17 responses

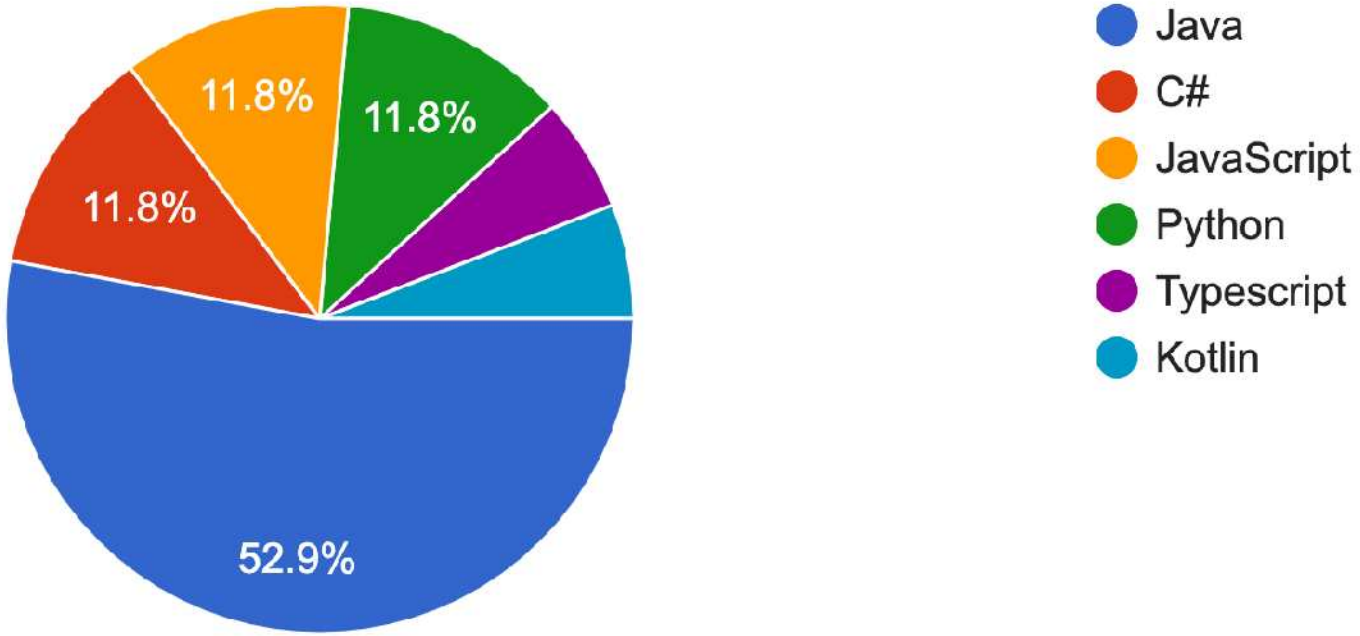


- Entwickler
- Architekt
- Product-Owner
- Technical Writer
- Professor Medizintechnik mit einem SW-Schwerpunkt
- Technical Lead
- Requirements Engineer
- Agile Coach

Wer seid Ihr?

Was ist Deine hauptsächlich verwendete Programmiersprache?

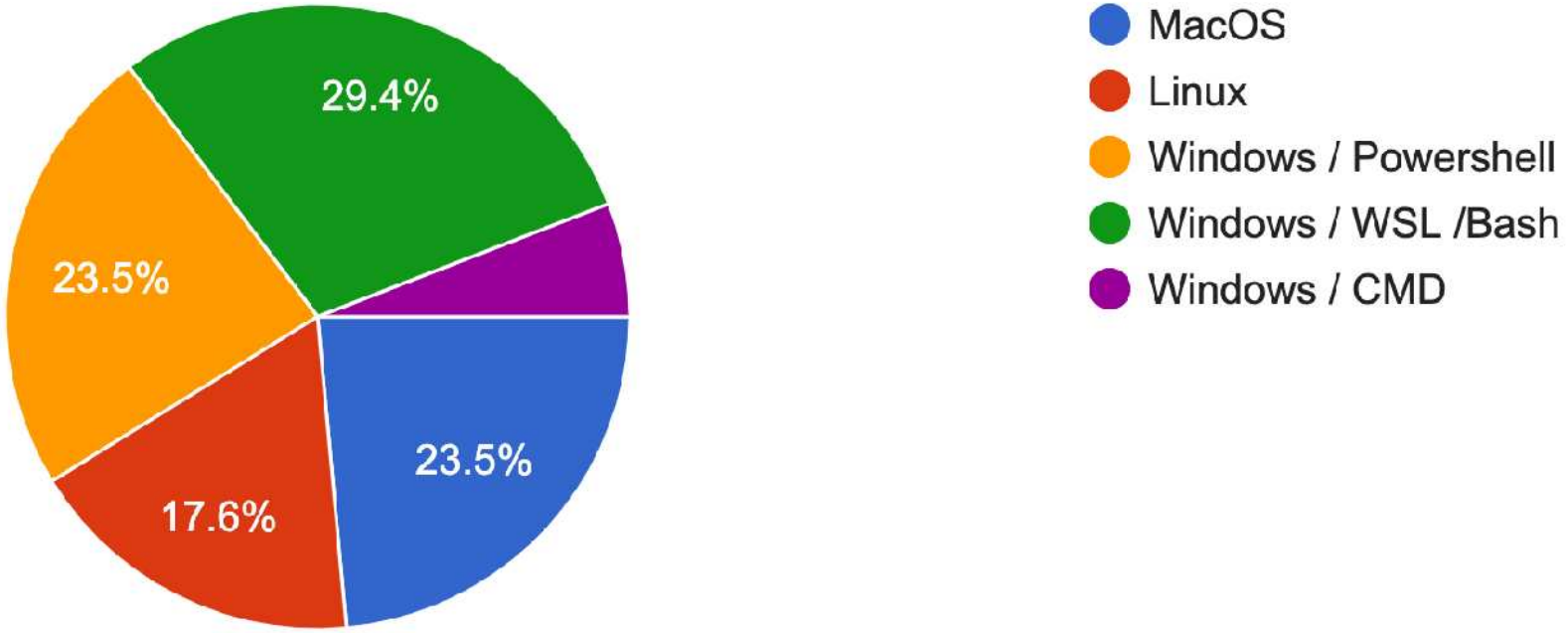
17 responses



Wer seid Ihr?

Welches Kombination Betriebssystem/Terminal nutzt Du?

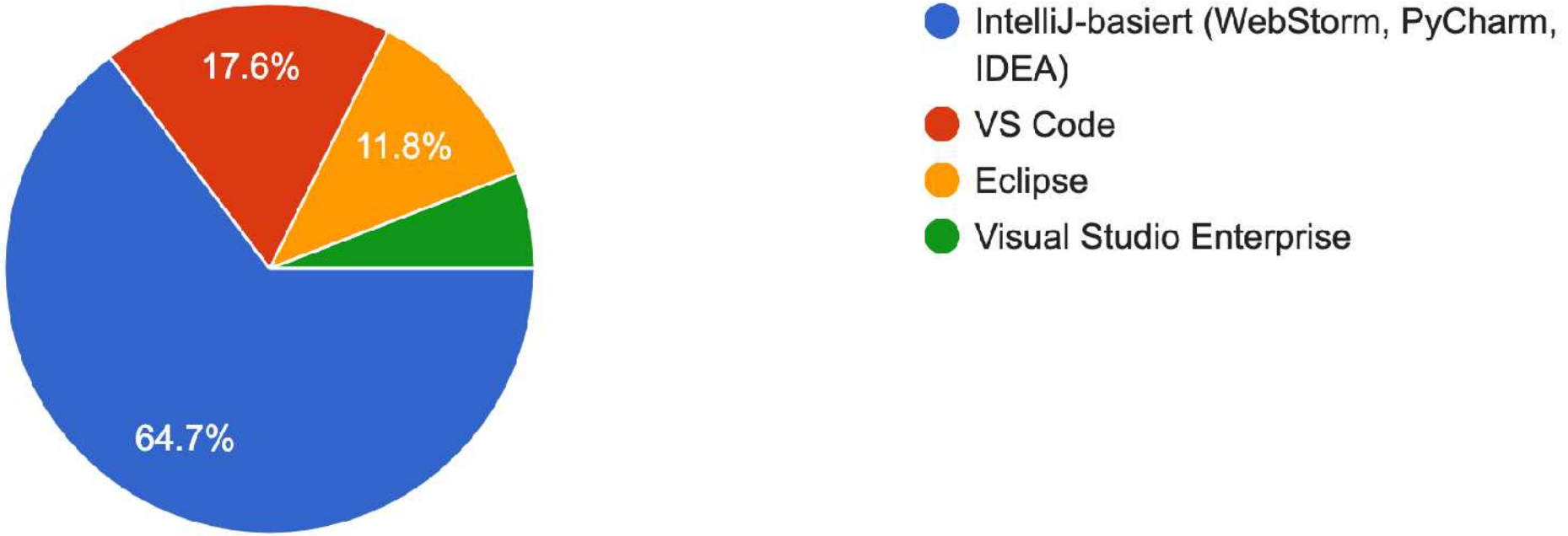
17 responses



Wer seid Ihr?

Welche IDE nutzt Du am liebsten?

17 responses



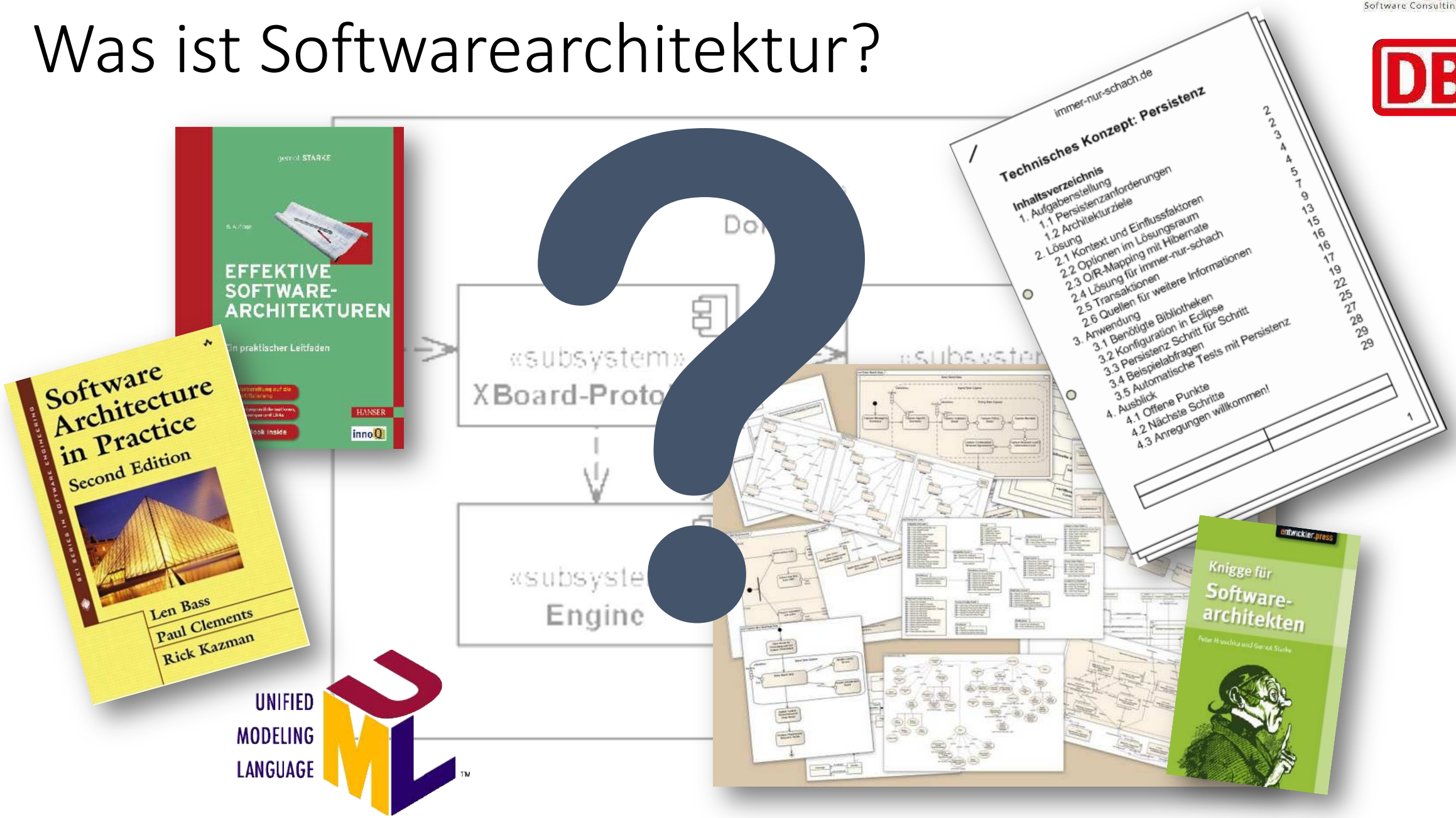
Agenda

- Einführung
- Documentation as Code
- docToolchain
- Architektur dokumentieren
- Praktische Umsetzung
- Weiterführende Themen
- Fazit und Ausblick

Agenda

- **Einführung**
- Documentation as Code
- docToolchain
- Architektur dokumentieren
- Praktische Umsetzung
- Weiterführende Themen
- Fazit und Ausblick

Was ist Softwarearchitektur?



Experten zu: Was ist Softwarearchitektur?

> *"… not all design is architecture. Architecture represents the **significant design decisions** that **shape a system**, where significant is measured by cost of change."*
>
> (Grady Booch)

> *"Softwarearchitecture is about **the important stuff**, **whatever that is**."*
>
> (Ralph Johnson)

> *"Software architecture is the **set of design decisions** which, if **made incorrectly**, may cause your **project to be cancelled**."*
>
> (Eoin Woods)

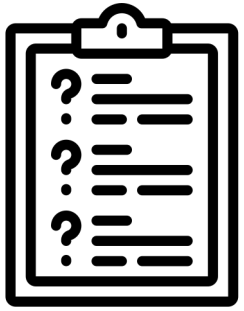
Was ist Softwarearchitektur?

Softwarearchitektur :=

$\sum$ wichtige Entscheidungen

wichtig :=

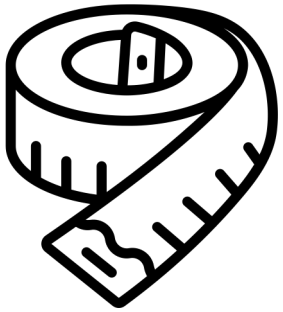
- fundamental (betrifft viele)
- im weiteren Verlauf nur schwer zu ändern
- entscheidend für den Erfolg des Softwaresystems



Anforderungen klären

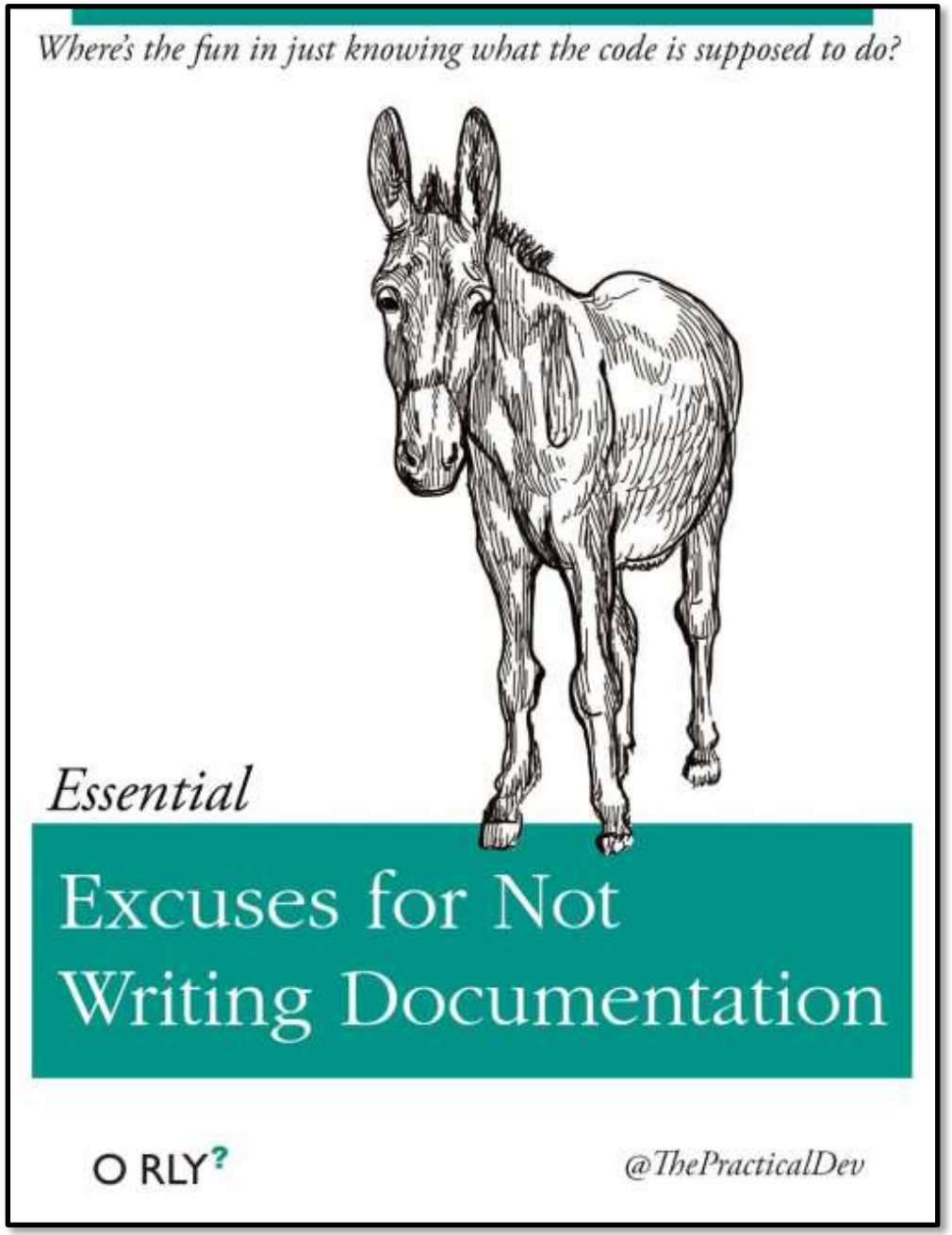
Architektur entwerfen

Architektur kommunizieren

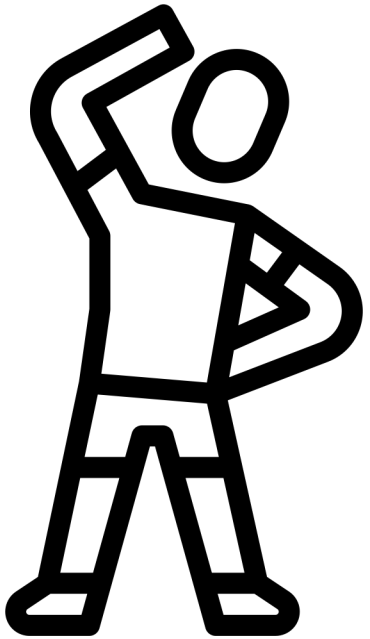


Architektur bewerten

aus: Effektive Softwarearchitekturen (G. Starke)

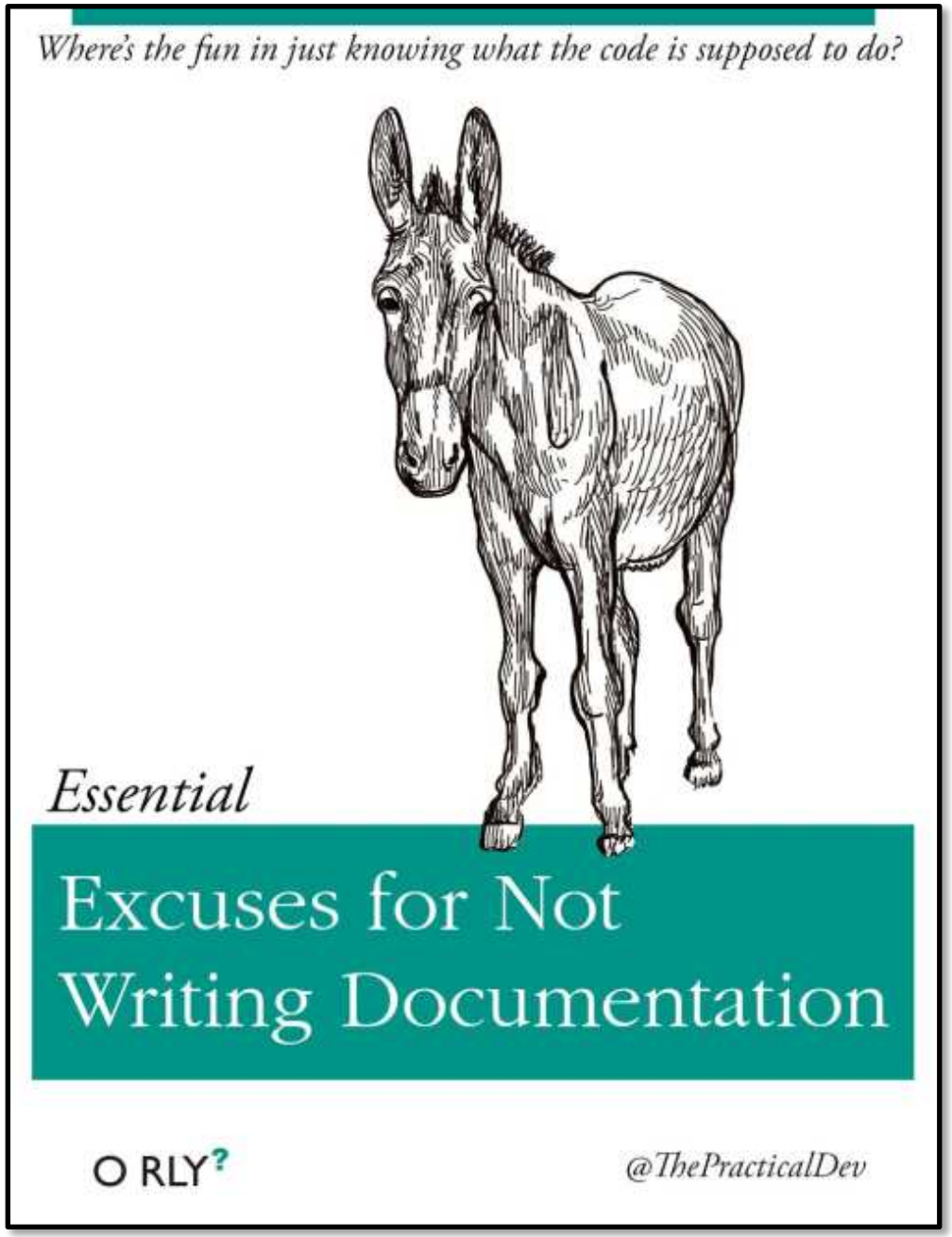


Grafik von The Practical Dev (CC0 Public Domain Lizenz)



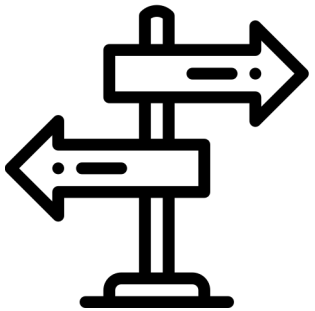
Bitte folgt dem Link für eine kleine Übung (01)

https://tinyurl.com/etffm-docs

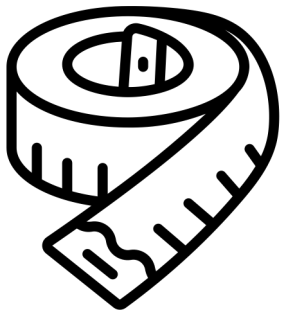


Grafik von The Practical Dev (CC0 Public Domain Lizenz)

Entwurfsunterstützung

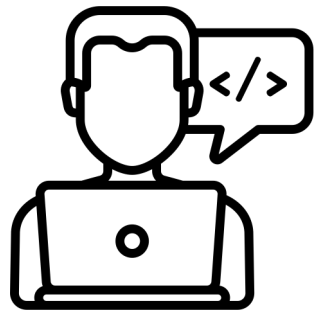


Frage nach dem Warum



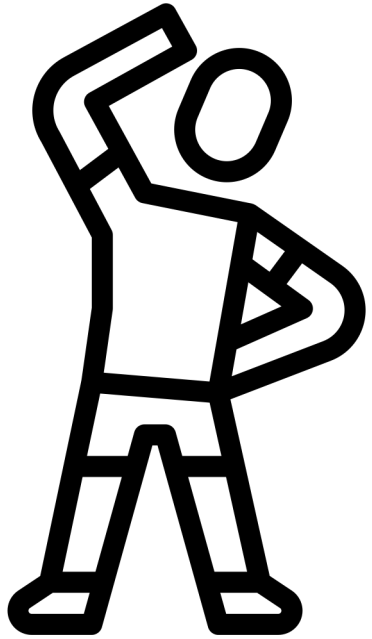
Bewertbarkeit

Kommunikation



Neue Mitarbeiter

Hauptsache, du machst es nicht mit Word!

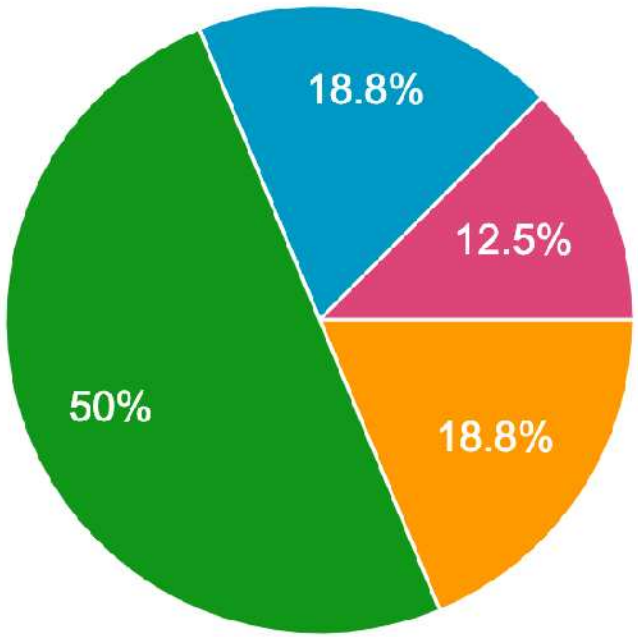


Bitte folgt dem Link für eine kleine Übung (02)

https://tinyurl.com/etffm-docs

Mit welchem Tool schreibst Du bislang Dokumentation?

16 responses



- Dokumentation? Das macht irgendwer anderes...
- Dokumentation? Die schreibt unser Technical Writer
- Word erleichtert mir die Dokumentation
- Confluence lässt uns gemeinsam an Doku arbeiten
- PowerPoint, damit ich sie auch präsen…
- Mein Code ist die ganze Doku, die ich…
- Die Stories in Jira bilden zusammen di…

Liest sowieso keiner!

Ungeeignete Werkzeuge

Veraltet

Medienbruch

Handarbeit

Dateiablage

Textwüsten

Redundanzen

Mission Statement (für diesen Vortrag)

- Klären, was man von der **Softwarearchitektur dokumentieren** sollte und wie uns **arc42** bei der **Strukturierung hilft**.
- Umsetzung des **Documentation-as-Code** Ansatzes mit **docToolchain** und dem ganz neuen **Wrapper dtcw** für die einfache Installation auf der **Kommandozeile**.
- Überblick über die sehr **effektiven** Funktionen von **AsciiDoctor** und dessen **mächtigen Integrationsmöglichkeiten** im Umfeld von Architekturdokumentation vermitteln

Agenda

- Einführung
- **Documentation as Code**
- docToolchain
- Architektur dokumentieren
- Praktische Umsetzung
- Weiterführende Themen
- Fazit und Ausblick

Unser täglich Entwickler-Brot

- Plain-Text
- Entwicklungsumgebung
- Kommandozeilen-werkzeuge
- **Versionsverwaltung**

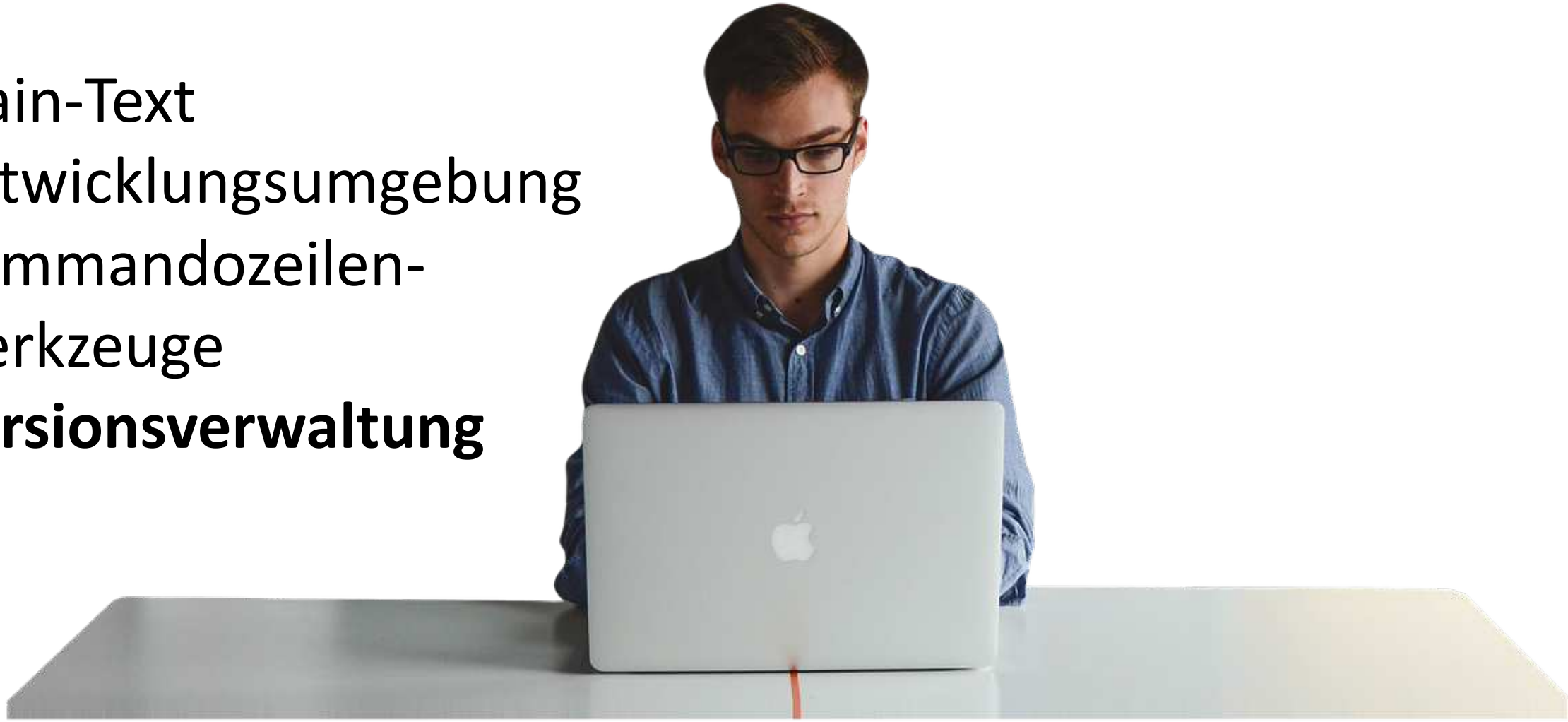
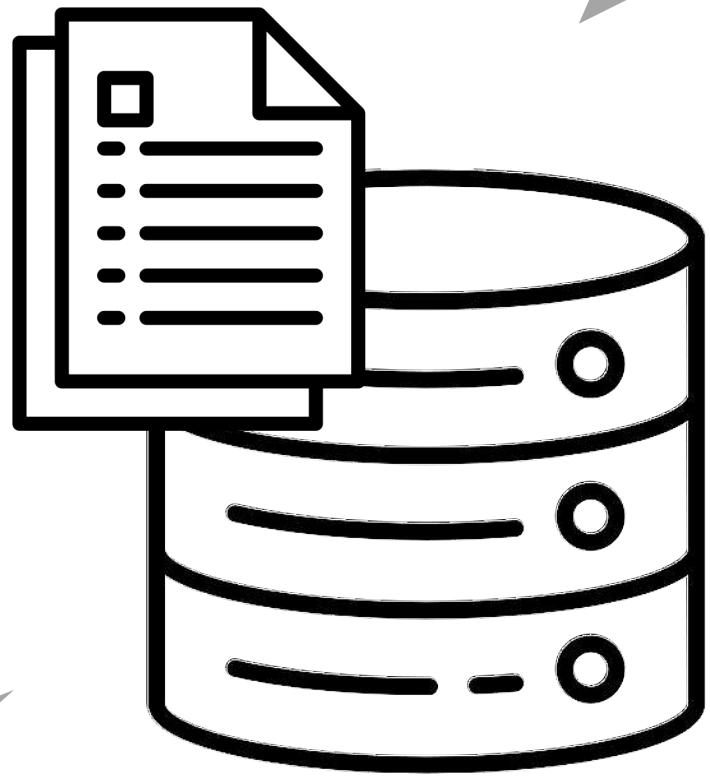


Foto von geralt: https://pixabay.com/de/unternehmer-start-start-up-karriere-696976/ (CC0 Public Domain Lizenz)

Documentation-as-Code

Nähe zum Sourcecode

Flexible Ausgabe



Ablage im Repo

Generierung/ Automatisierung

Versionier-/Diffbar

Teil des Build-Prozesses

Synchrone Auslieferung

Offlinefähig

Michael Simons
@rotnroll666

Folge ich

Think I'm more a MarkDown person than AsciiDoc. The results are great, but MarkDown needs less concentration to write.

Markdown

- Markdown
  - Toller Standard für einfache Auszeichnungen
  - Features

Span Elements
- Links
- Emphasis
- Code
- Images

Block Elements
- Paragraphs and Line Breaks
- Headers
- Blockquotes
- Lists
- Code Blocks
- Horizontal Rules

TOC
Tables (Feature Rich)
Includes (Level-Offset)
PlantUML
Admonitions
Attributes
Anchors
Footnotes, Index, Glossary
Videos
Syntax Highlighting
Callouts
Math Rendering
Outputformats

Markdown

Wir brauchen eine Erweiterung!

Welche wählen wir?

- CommonMark
- CriticMarkup
- Discount
- DocFX
- ExtraMark
- Ghost's Markdown/Haunted Markdown
- GitHub Flavored Markdown
- GitLab Flavored Markdown (with login)
- Haroopad Flavored Markdown
- iA Writer's Markdown
- Kramdown
- Leanpub Flavored Markdown
- Litedown
- Lunamark
- Madoko
- Markdown
- Markdown 2
- Markdown Extra
- Markdown-it
- Markua
- Maruku
- MultiMarkdown
- Pandoc's Markdown
- PHP Markdown Extra Extended
- Python Markdown
- R Markdown
- Redcarpet
- Remarkable
- Rhythmus
- Scholarly Markdown
- Showdown
- StackOverflow's Markdown
- Taiga Markdown
- Trello's Markdown
- vfmd
- Xcode/Swift Playgrounds Markup

https://github.com/commonmark/commonmark-spec/wiki/markdown-flavors

Markdown einen Dialekt

Wir brauchen ~~eine Erweiterung~~!

Welche wählen wir?

Funktioniert dann unsere Toolchain noch?

Haben wir ein Editor-Preview?

Michael Simons
@rotnroll666

Folge ich

Think I'm more a MarkDown person than AsciiDoc. The results are great, but MarkDown needs less concentration to write.

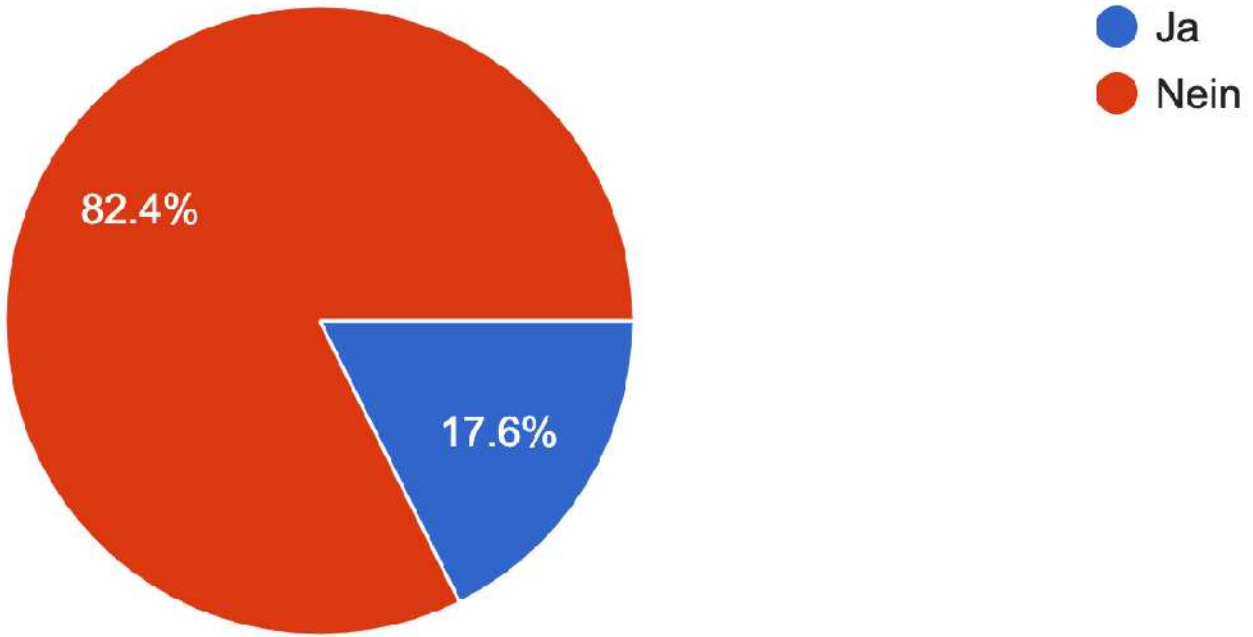
Ralf D. Müller
@RalfDMueller

Folgen

@rotnroll666 but the possibilities of AsciiDoc are better! Include code directly from the repository, render plantUML, subdocuments etc...

Kennst Du schon AsciiDoc?

17 responses



AsciiDoc / Asciidoctor

leistungsfähige Syntax für technische Dokumentation

in Ruby geschrieben

mit Opal nach JavaScript transpiliert

mit jRuby auf der JVM gewrapped

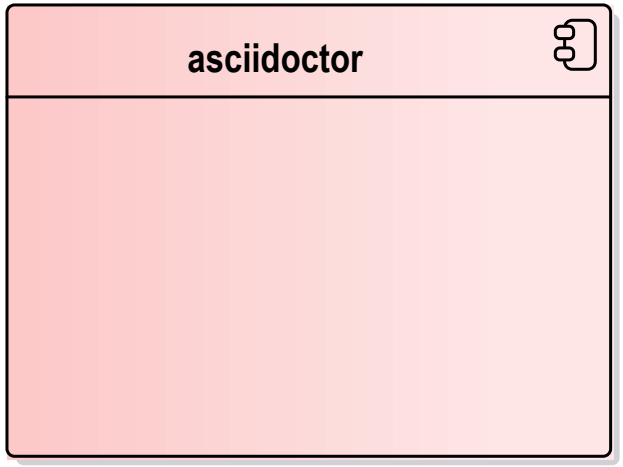
=> Keine Dialekte!

Continuous Documentation

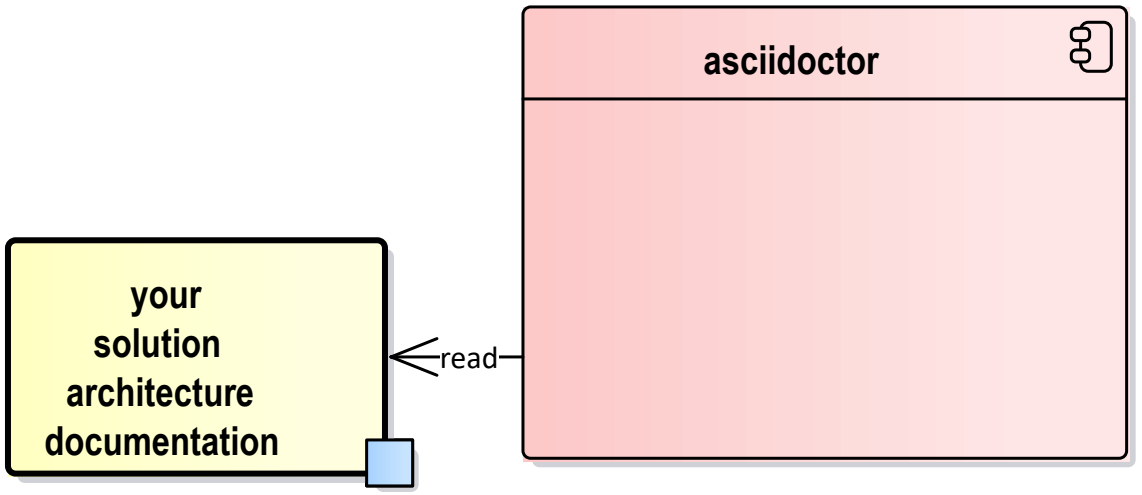
Agenda

- Einführung
- Documentation as Code
- **docToolchain**
- Architektur dokumentieren
- Praktische Umsetzung
- Weiterführende Themen
- Fazit und Ausblick

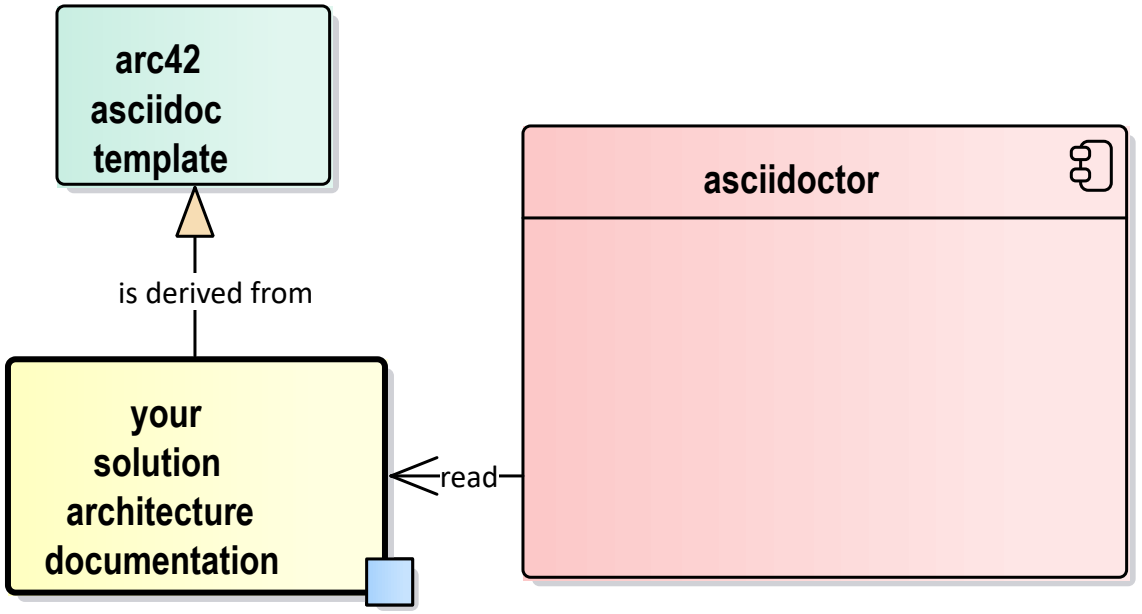
Basic Docs-as-Code with AsciiDoc



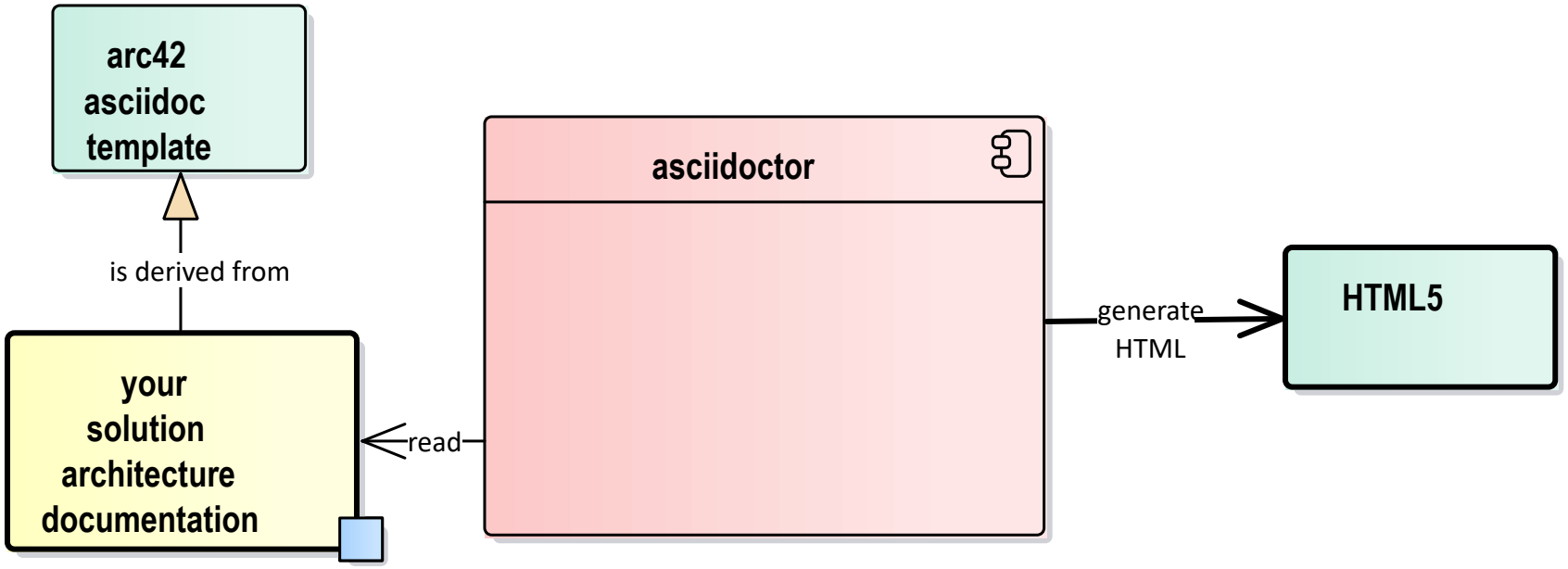
Basic Docs-as-Code with AsciiDoc



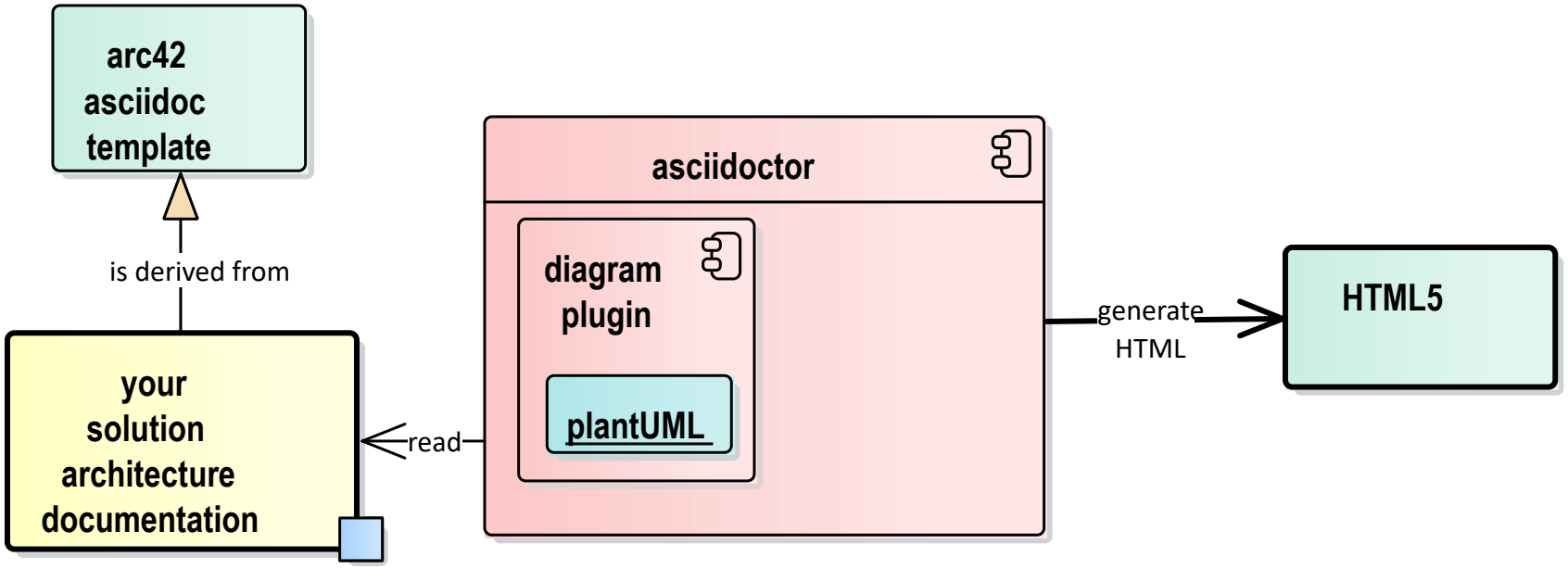
Basic Docs-as-Code with AsciiDoc



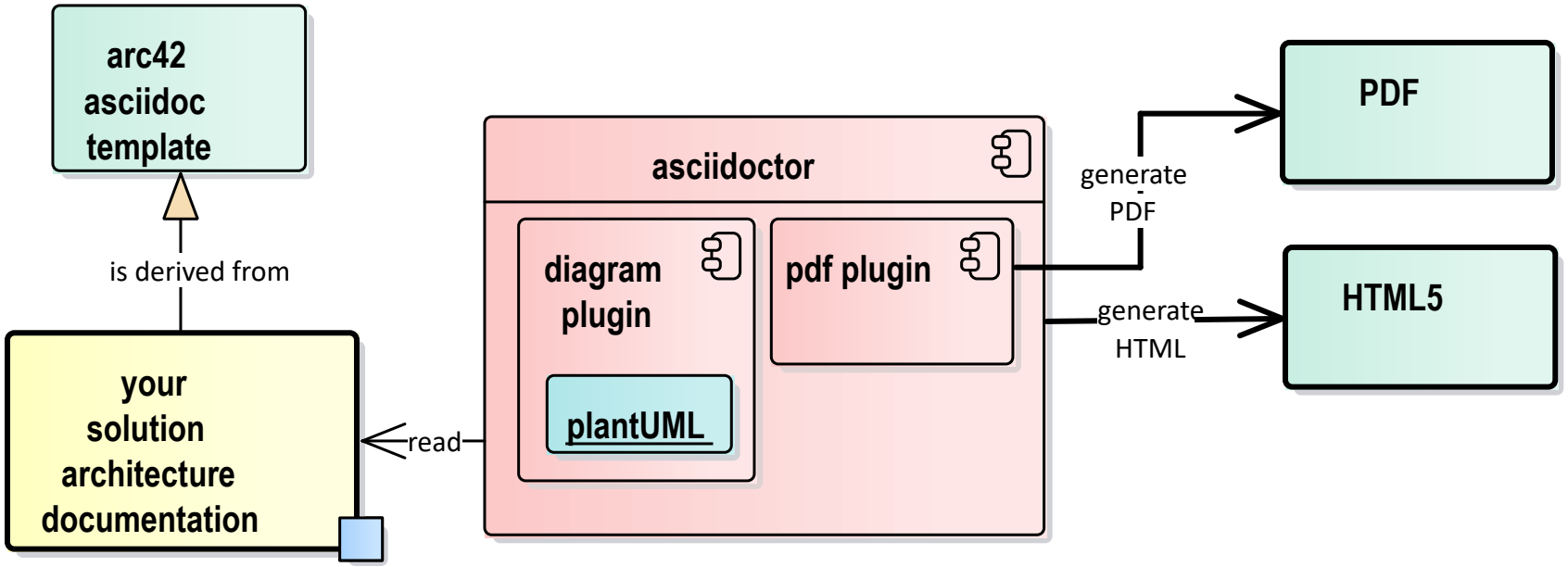
Basic Docs-as-Code with AsciiDoc



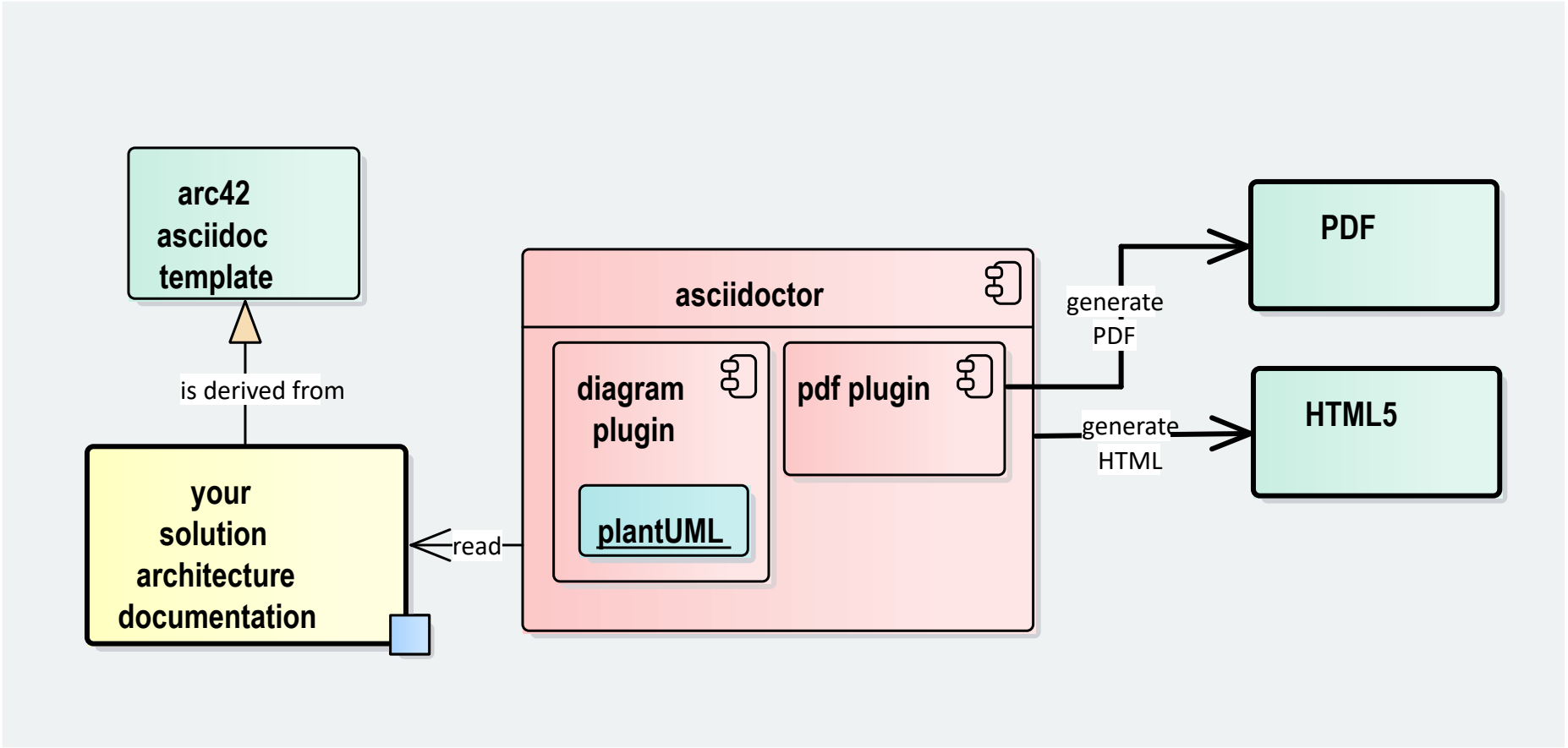
Basic Docs-as-Code with AsciiDoc



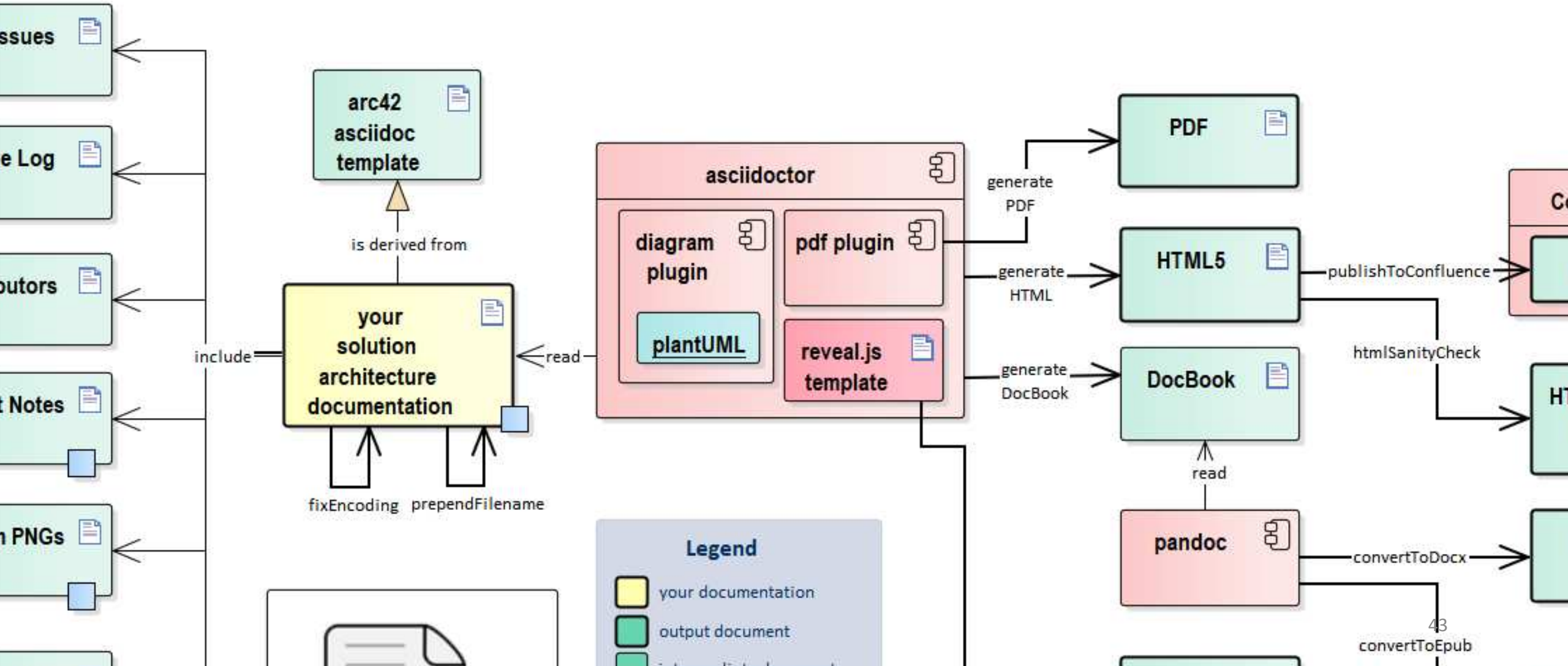
Basic Docs-as-Code with AsciiDoc



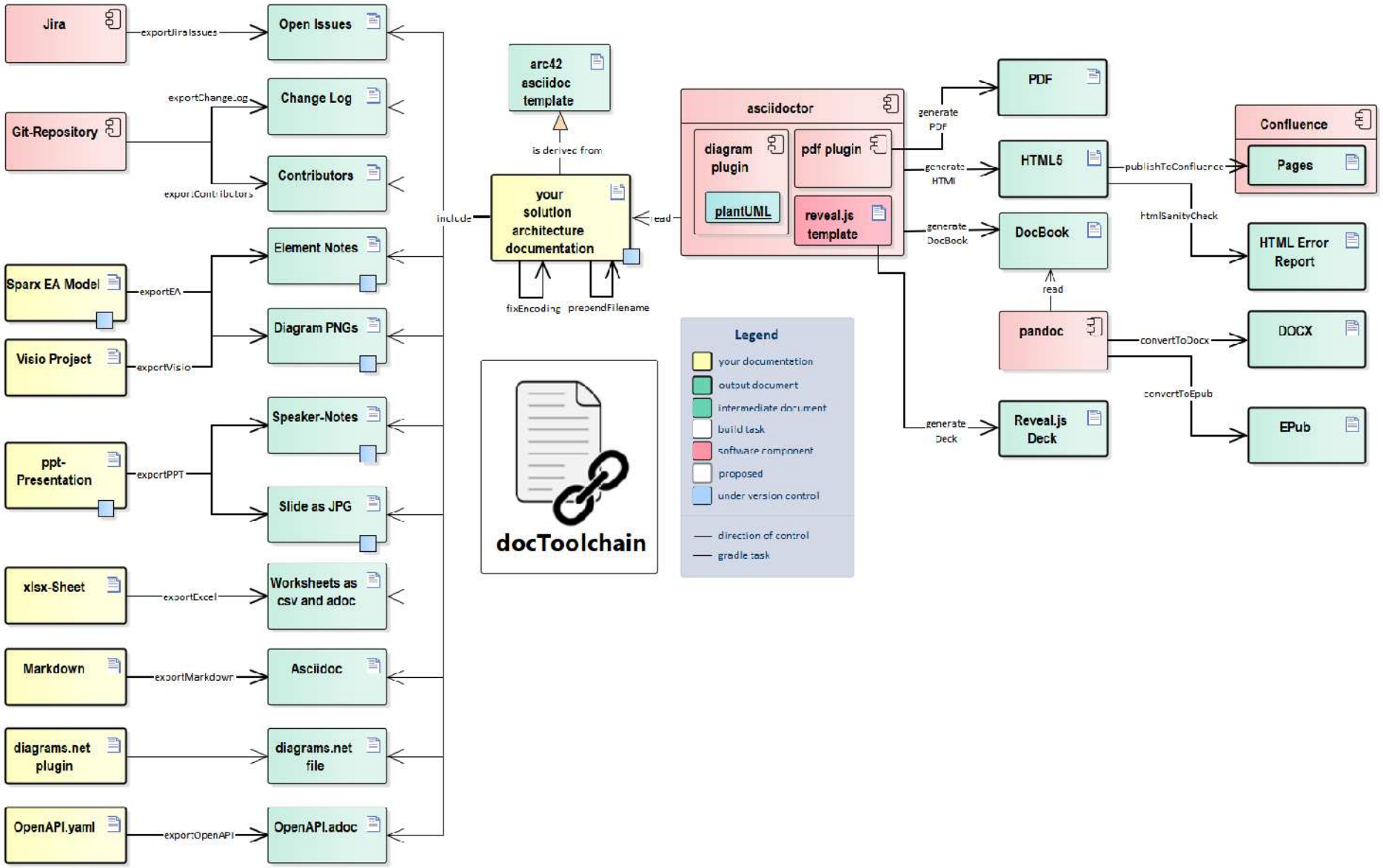
Basic Docs-as-Code with AsciiDoc & docToolchain



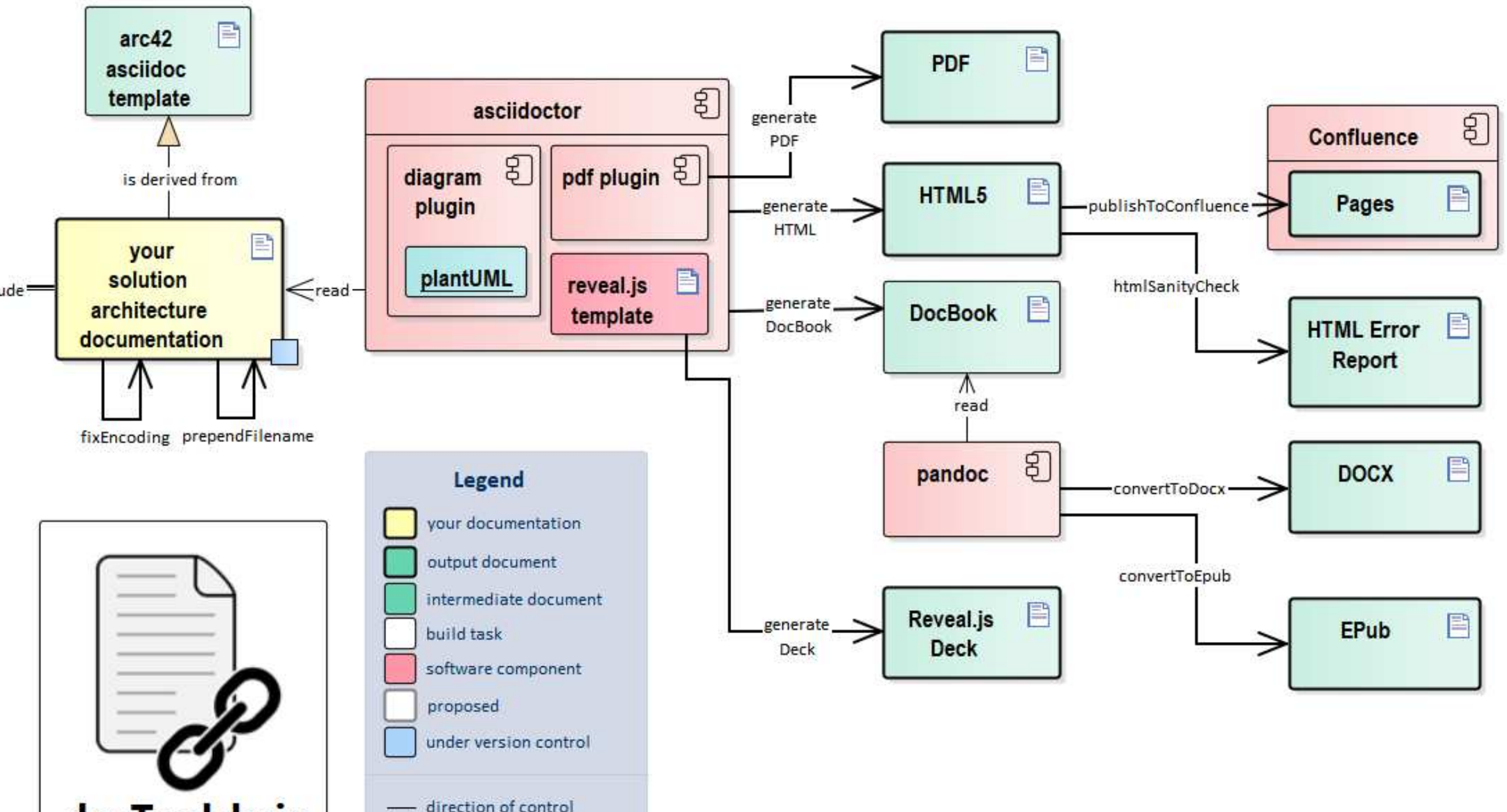
Basic Docs-as-Code with AsciiDoc & docToolchain



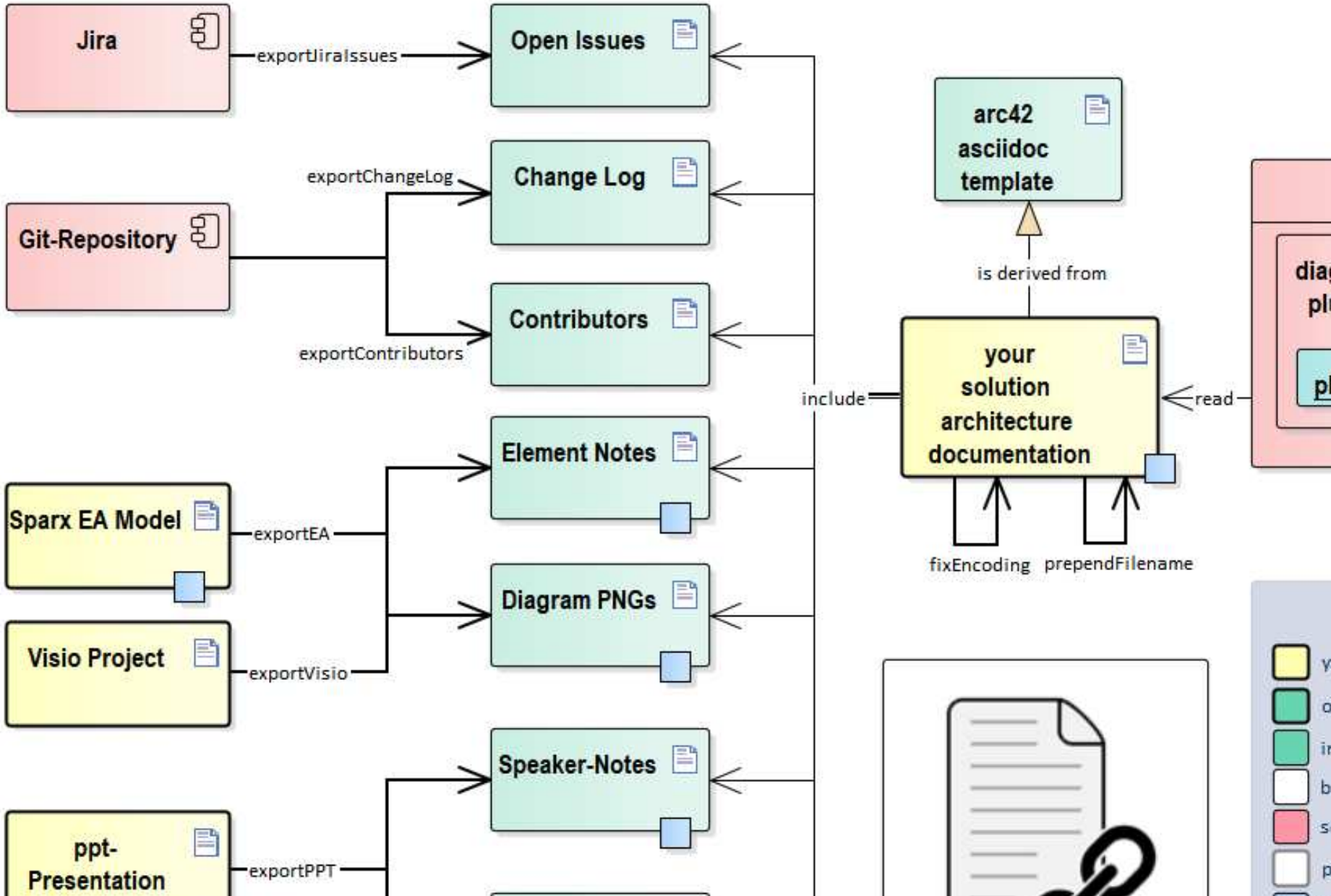
Basic Docs-as-Code with AsciiDoc & docToolchain



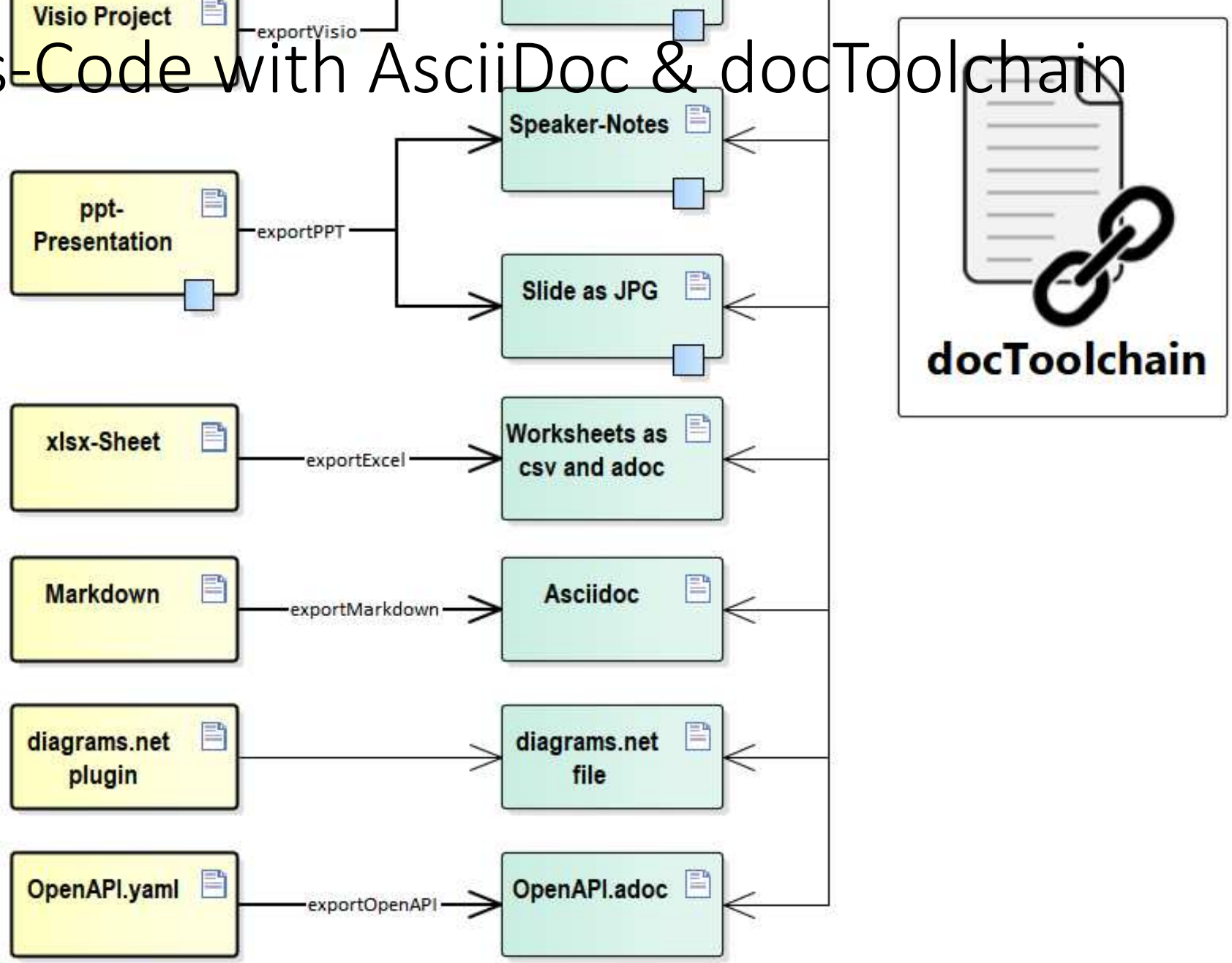
Basic Docs-as-Code with AsciiDoc & docToolchain



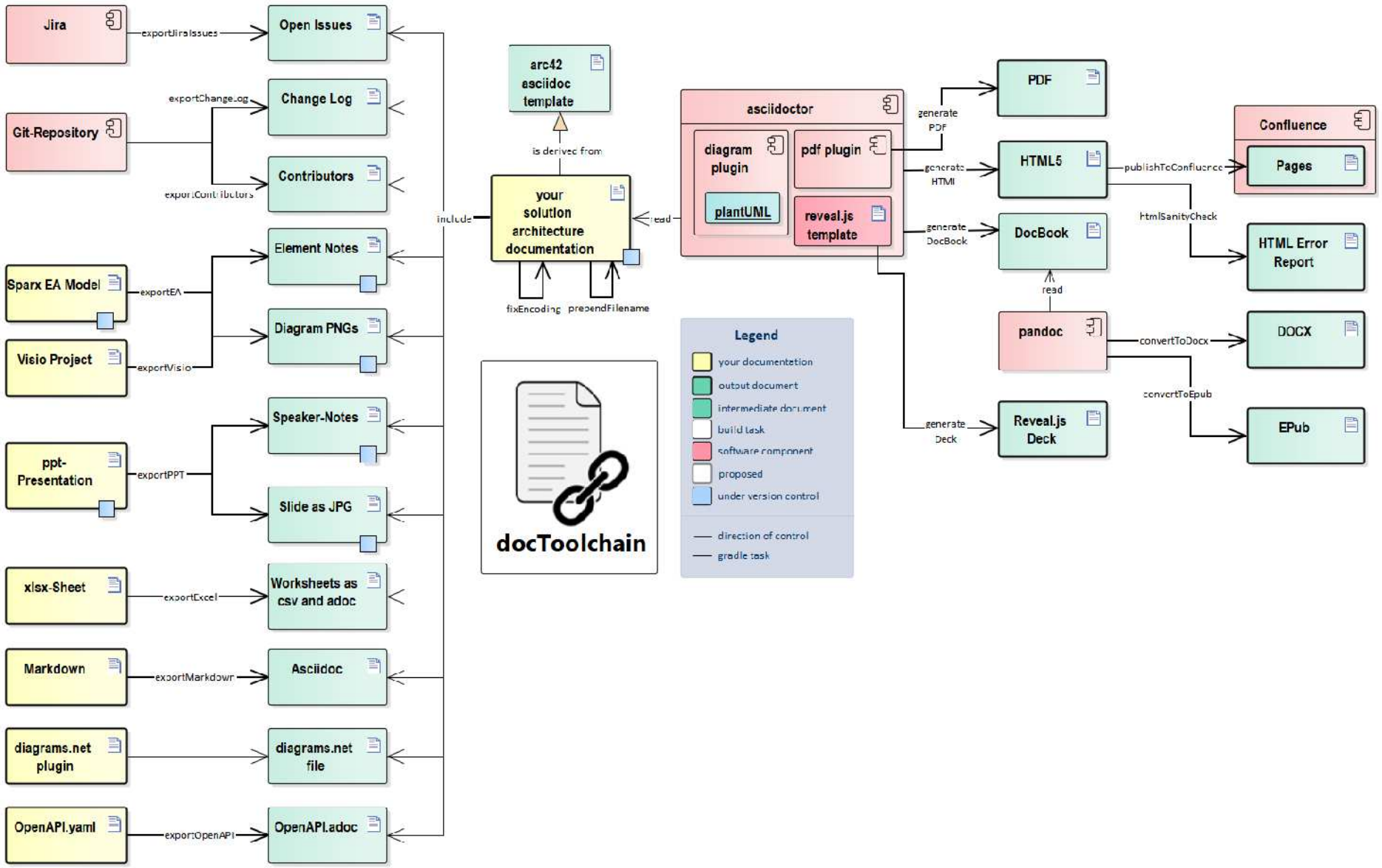
Basic Docs-as-Code with AsciiDoc & docToolchain



Basic Docs-as-Code with AsciiDoc & docToolchain



Basic Docs-as-Code with AsciiDoc & docToolchain



```
git clone
 git@github.com:docToolchain/workshop.git

mkdir solution

cd workshop/solution
```

```
curl -Lo dtcw doctoolchain.github.io/dtcw

wget doctoolchain.github.io/dtcw

curl –o dtcw.ps1
                    doctoolchain.github.io/dtcw.ps1


chmod +x dtcw
```

```
./dtcw tasks --group=doctoolchain

./dtcw downloadTemplate

./dtcw generateHTML
open ./build/html5/arc42/arc42.html

./dtcw generatePDF
open ./build/pdf/arc42/arc42.pdf
```

Kurze Pause bis 09:35 Uhr

Agenda

- Einführung
- Documentation as Code
- docToolchain
- **Architektur dokumentieren**
- Praktische Umsetzung
- Weiterführende Themen
- Fazit und Ausblick

Architekturdokumentation: Ziele.

Drei mögliche Ziele:

- Beim Entwurf der Architektur unterstützen
- Die Umsetzung und Weiterentwicklung des Systems leiten
- Die Architektur nachvollziehbar und bewertbar machen

7 Regeln für gute Dokumentation

„Documenting Software Architectures: Views and Beyond“
Clements, et.al, 2. Auflage 2010

1. Schreibe aus Sicht des Lesers
2. Vermeide unnötige Wiederholungen
3. Vermeide Mehrdeutigkeiten
    3. a) Erkläre Deine Notation
4. Verwende eine Standardstrukturierung
5. Halte Begründungen für Entscheidungen fest
6. Halte Dokumentation aktuell, aber auch nicht zu aktuell
7. Überprüfe Dokumentation auf ihre Gebrauchstauglichkeit

7 Regeln für gute Dokumentation

„Documenting Software Architectures: Views and Beyond“
Clements, et.al, 2. Auflage 2010

1. Schreibe aus Sicht des Lesers
2. Vermeide unnötige Wiederholungen
3. Vermeide Mehrdeutigkeiten
    3. a) Erkläre Deine Notation
4. **Verwende eine Standardstrukturierung**
5. Halte Begründungen für Entscheidungen fest
6. Halte Dokumentation aktuell, aber auch nicht zu aktuell
7. Überprüfe Dokumentation auf ihre Gebrauchstauglichkeit

„Abheften“ in arc42

1. Einführung und Ziele
1.1 Aufgabenstellung
1.2 Qualitätsziele
1.3 Stakeholder

2. Randbedingungen
2.1 Technische Randbedingungen
2.2 Organisatorische Randbedingungen
2.3 Konventionen

3. Kontextabgrenzung
3.1 Fachlicher Kontext
3.2 Technischer- oder Verteilungskontext

4. Lösungsstrategie

5. Bausteinsicht
5.1 Ebene 1
5.2 Ebene 2
...

6. Laufzeitsicht
6.1 Laufzeitszenario 1
6.2 Laufzeitszenario 2
...

7. Verteilungssicht
7.1 Infrastruktur Ebene 1
7.2 Infrastruktur Ebene 2
...

8. Konzepte
8.1 Fachliche Strukturen und Modelle
8.2 Typische Muster und Strukturen
8.3 Persistenz
8.4 Benutzungsoberfläche
...

9. Entwurfsentscheidungen
9.1 Entwurfsentscheidung 1
9.2 Entwurfsentscheidung 2
...

10. Qualitätsszenarien
10.1 Qualitätsbaum
10.2 Bewertungsszenarien

11. Risiken

12. Glossar

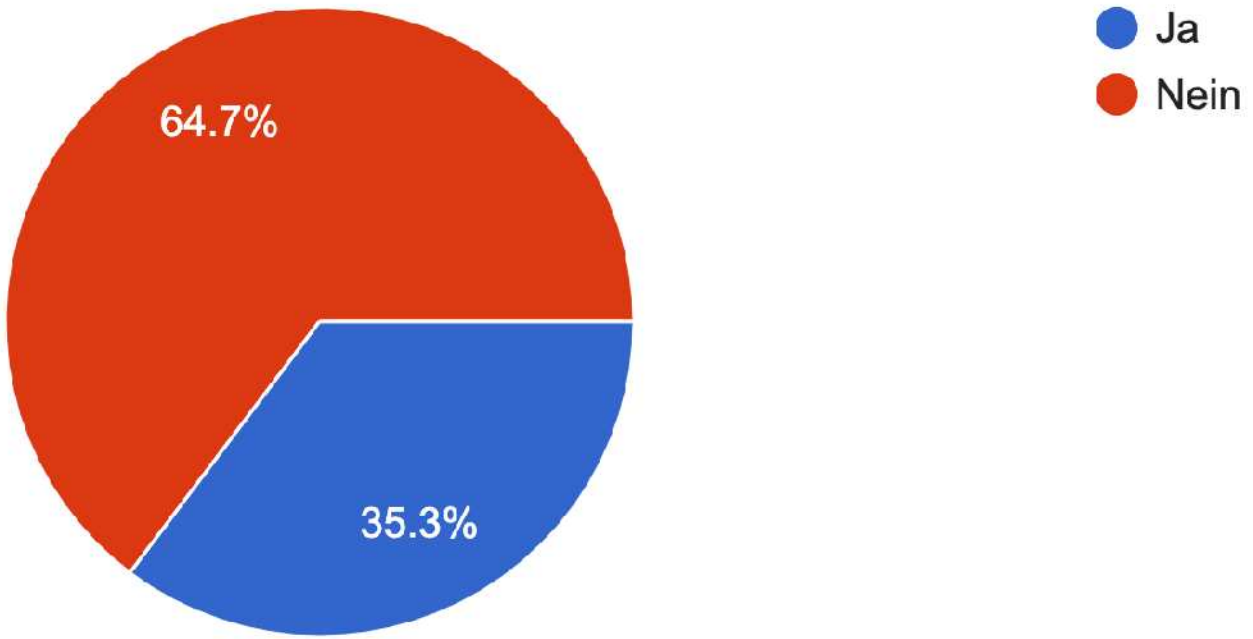
Dr. Peter Hruschka
http://www.peterhruschka.eu/

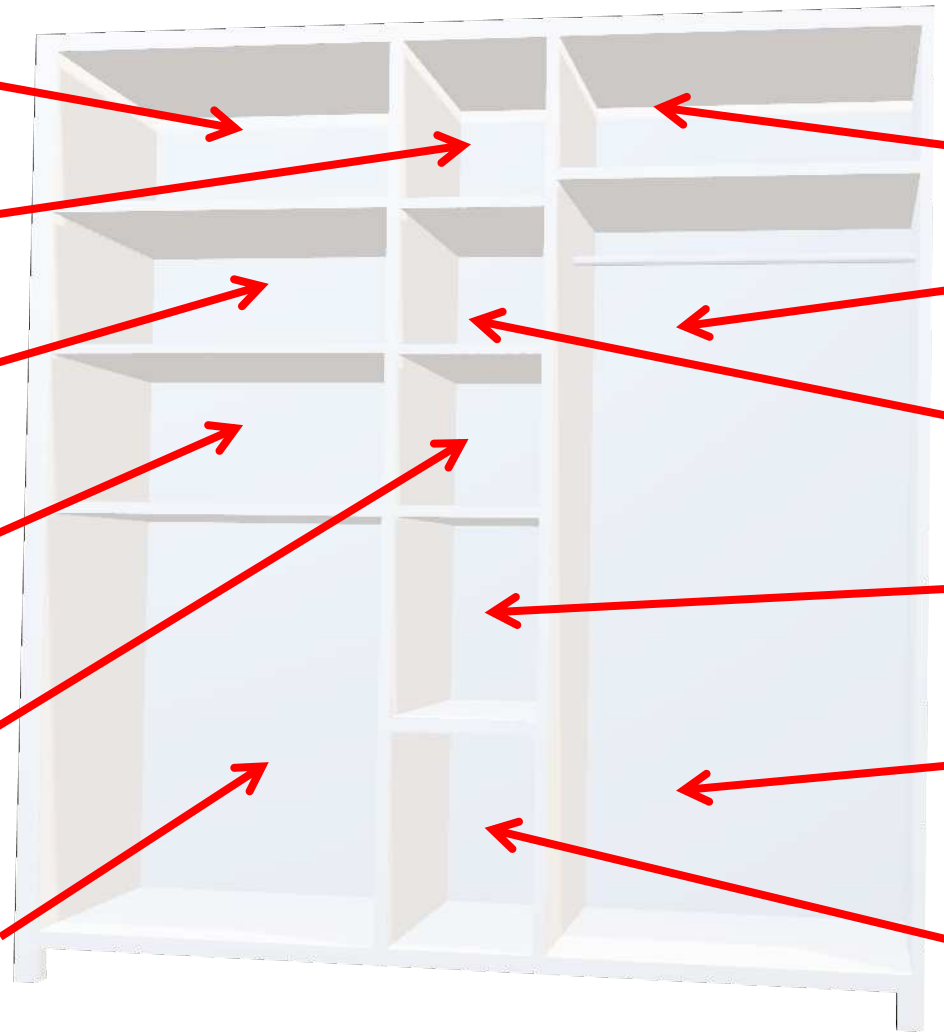
Dr. Gernot Starke
http://gernotstarke.de/

➔https://arc42.de/

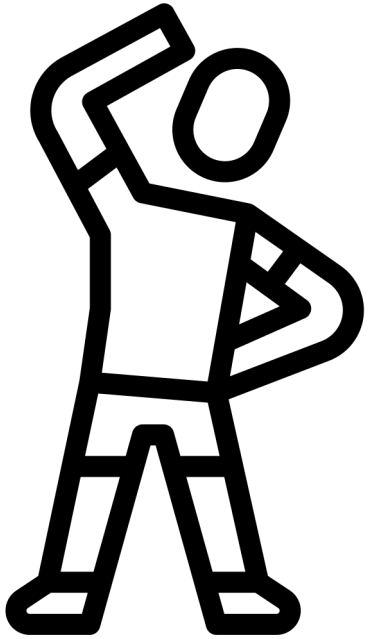
Kennst Du schon arc42?

17 responses





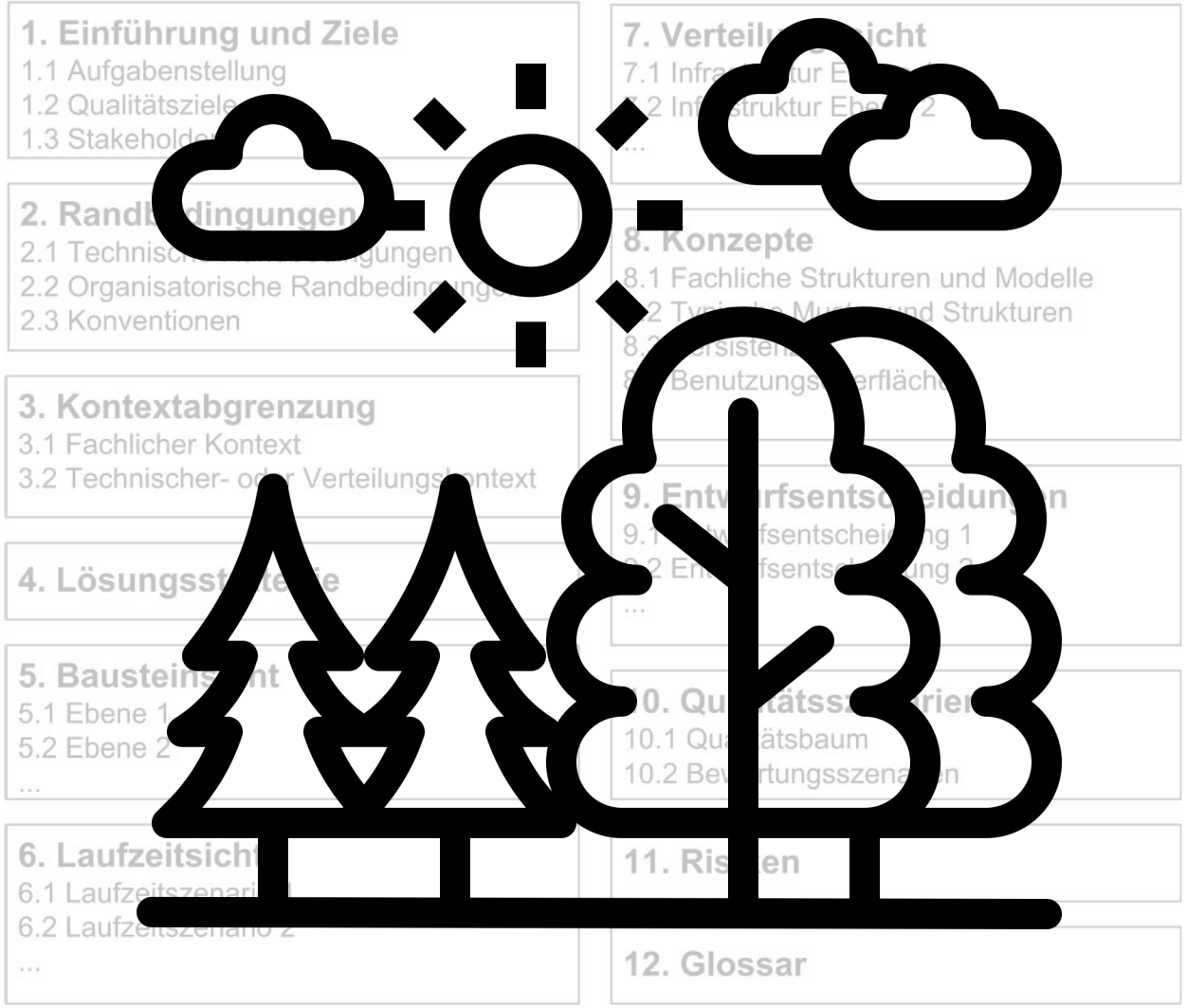
1. Requirements & Goals
2. Constraints
3. Scope & Context
4. Solution Strategy
5. Building Block View
6. Runtime View
7. Deployment View
8. Crosscutting Concepts
9. Decisions
10. Quality Scenarios
11. Risks & Tech Debt
12. Glossary



Bitte folgt dem Link für eine kleine Übung (03)

https://tinyurl.com/etffm-docs

Wald vor lauter Bäumen …



WAS sollen wir bauen?

Was wollen wir erreichen?
Wozu ist es da?
Wem nützt es?
Was soll es tun?
Was braucht es nicht tun?
Woran halten wir uns?
Was soll es exzellent können?

ANFORDERUNGEN

Architekturüberblick

Aufgabe

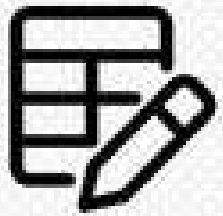
- Mission Statement
- Kontextabgrenzung

Einflüsse

- Rahmenbedingungen
- Qualitätsziele

Lösungsansätze

- Architekturstil
- Technologie-Stack
- Konzepte
- Prinzipien
- Zerlegung
- ....



Lösungsstrategie (plus Überblicksbild) als Brücke

WIE sieht die Lösung aus?

Wie erreichen wir das?
Welchen Mustern folgen wir?
Was verwenden wir?
Was leitet uns?
Woraus besteht es?
Wie ist es strukturiert?

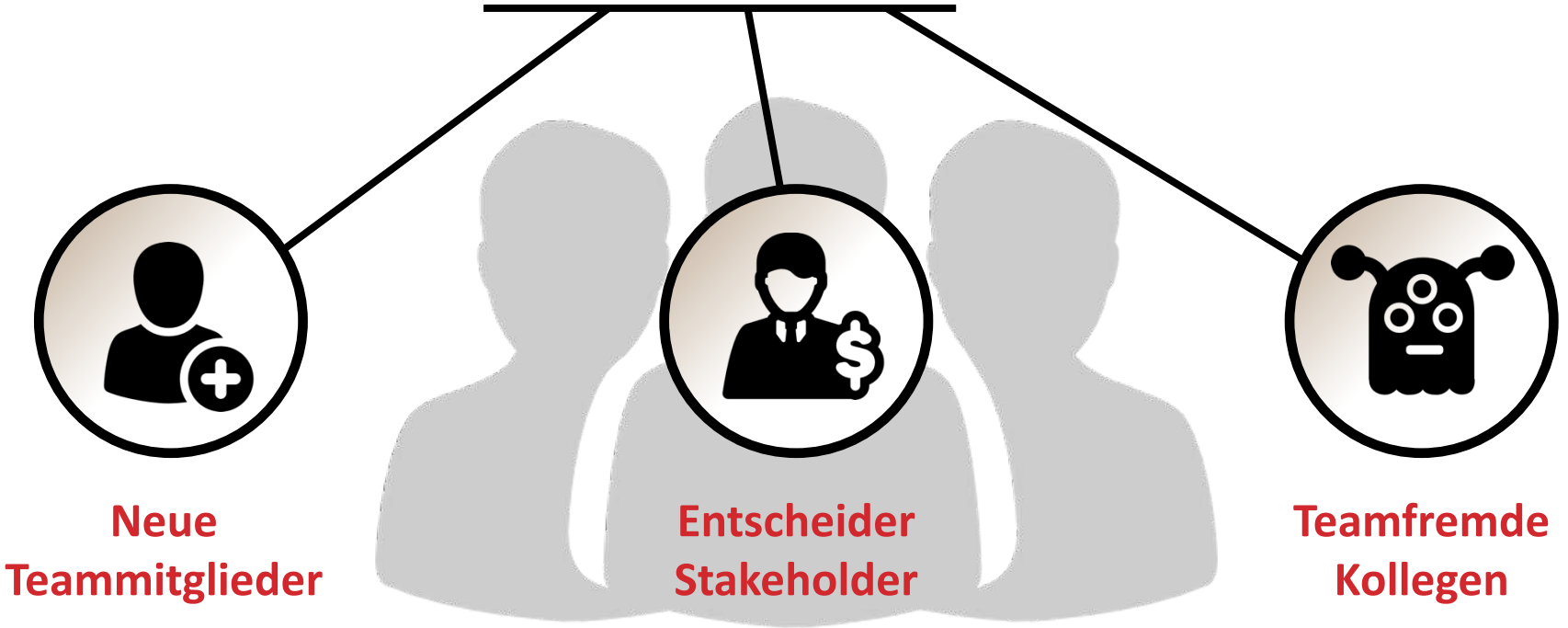
ENTSCHEIDUNGEN

embarc.de

Softwarearchitektur auf dem Bierdeckel

Was ist ein Architekturüberblick?

Ein Architekturüberblick macht die zentralen Lösungsansätze Eurer Softwarearchitektur für Außenstehende nachvollziehbar.




Java in a Nutshell
SYMANTEC
VisualCafé
SYMANTEC.
JDK™ 1.1 & JDK™ 1.2
Windows NT
Windows 95/98
Leichte und intuitive Anwendungsentwicklung in 100% Java™
• Zwei-Wege-Programmierung mit Drag-and-Drop
• Interaktionsassistent zur Bearbeitung der Beziehungen zwischen JavaBeans und Komponenten
• 100+ JavaBean-Komponenten
• Java Code-Hilfe
• JFC/Swing-Unterstützung
• Plug-in beliebiger JDKs
• Integrierter Editor, Debugger und Compiler
VisualCafé
DER SCHNELLE WEG, UM JAVA ZU LERNEN
Version 3
Standard Edition
Enterprise
JAVABEAN

Mission Statement

Die **Corona-Warn-App** ist eine App, die hilft, **Infektionsketten** des SARS-CoV-2 (COVID-19-Auslöser) in Deutschland **nachzuverfolgen** und zu **unterbrechen**.

Die App basiert auf Technologien mit einem **dezentralisierten Ansatz** und informiert Personen, wenn sie mit einer infizierten Person in Kontakt standen.

Transparenz ist von entscheidender **Bedeutung**, um die Bevölkerung zu schützen und die **Akzeptanz zu erhöhen**.

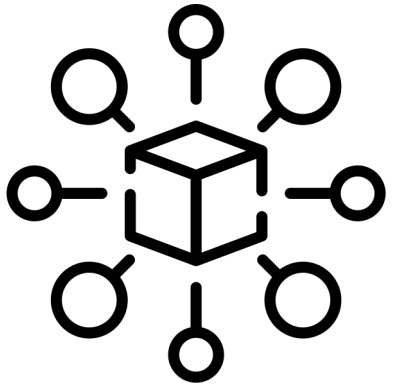
Quelle: https://www.coronawarn.app/de/

Wie funktioniert die App?

https://www.bundesregierung.de/breg-de/themen/corona-warn-app/corona-warn-app-erklaerfilm-1758828

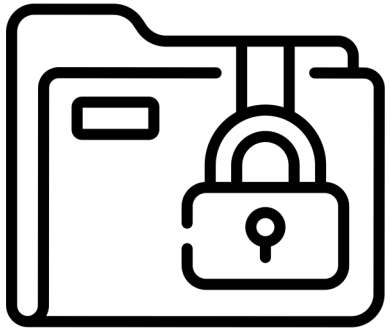
Was passiert mit den Daten?

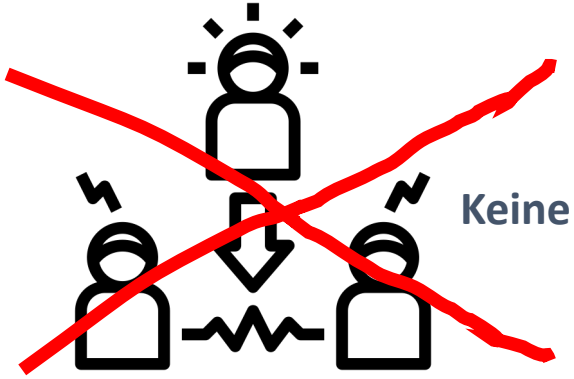
Keine Anmeldung



Dezentrale Speicherung

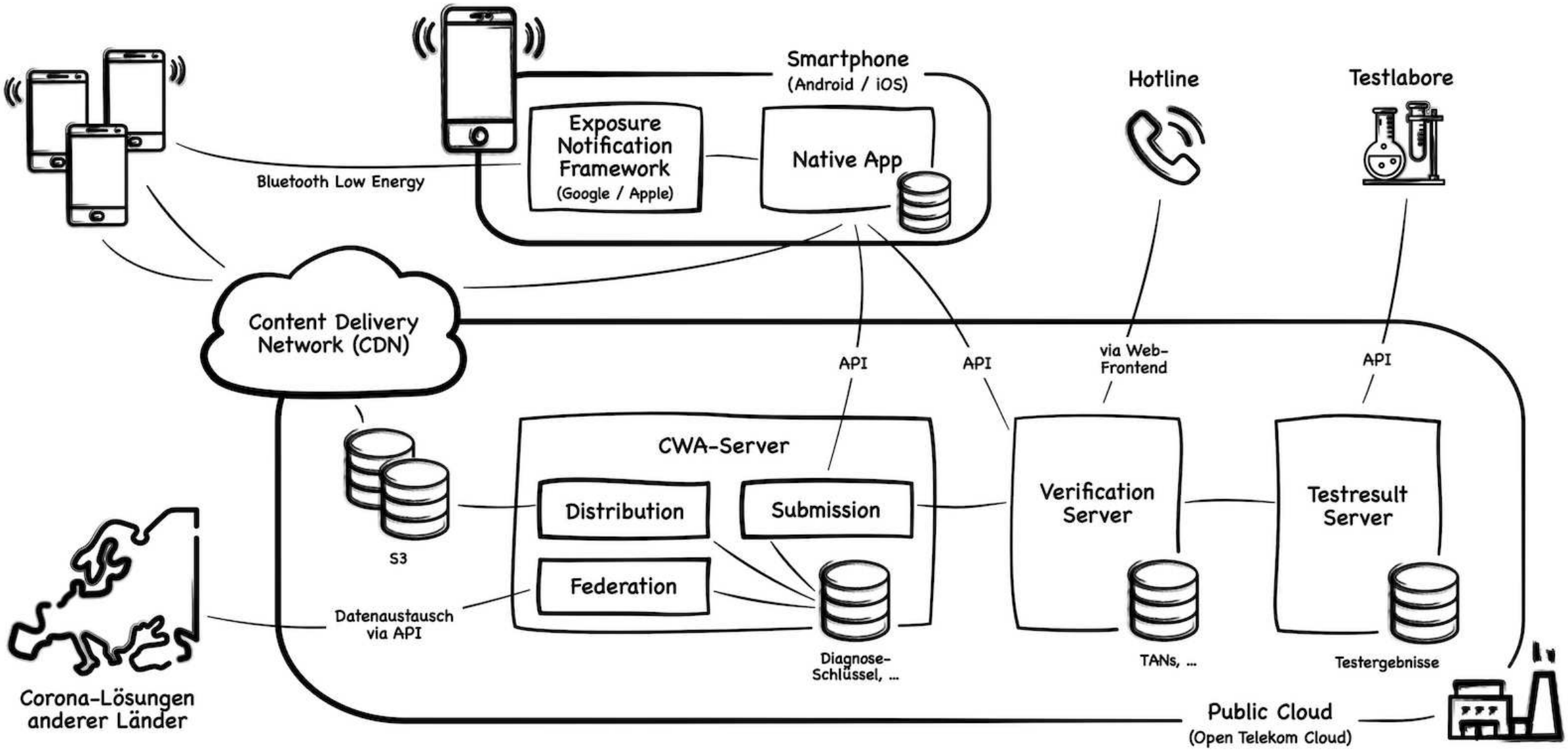
Keine Rückschlüsse auf persönliche Daten





Keine Einsicht für Dritte

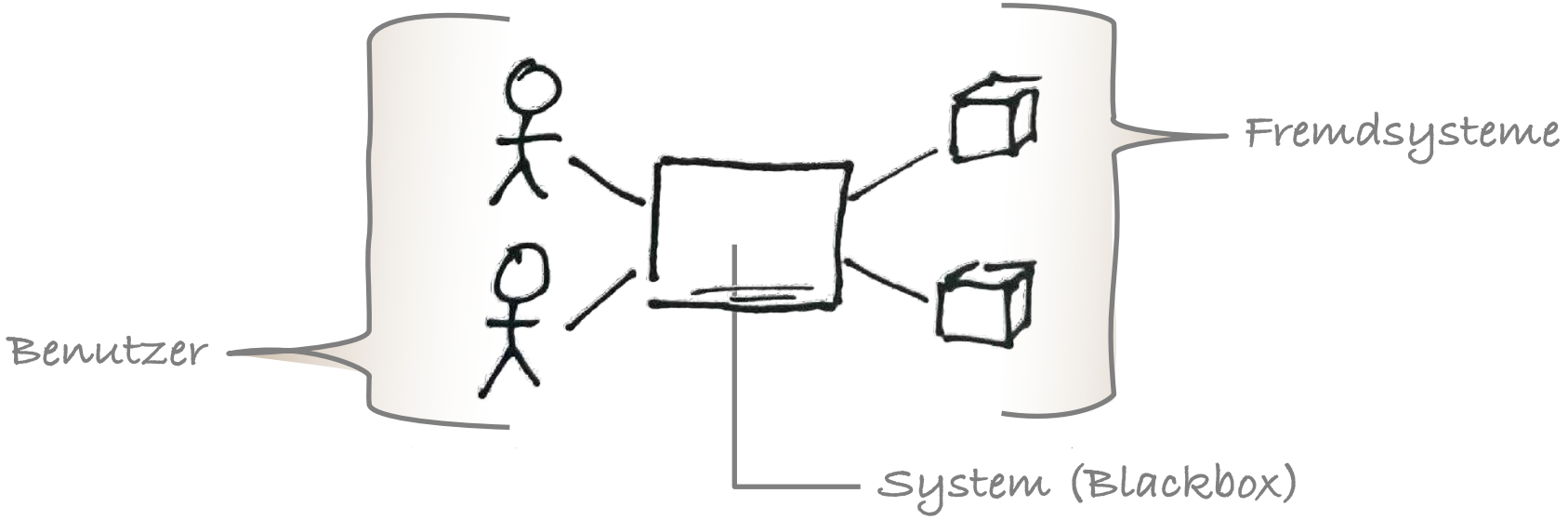
Architektur CWA: Informelles Überblicksbild



Kontextabgrenzung

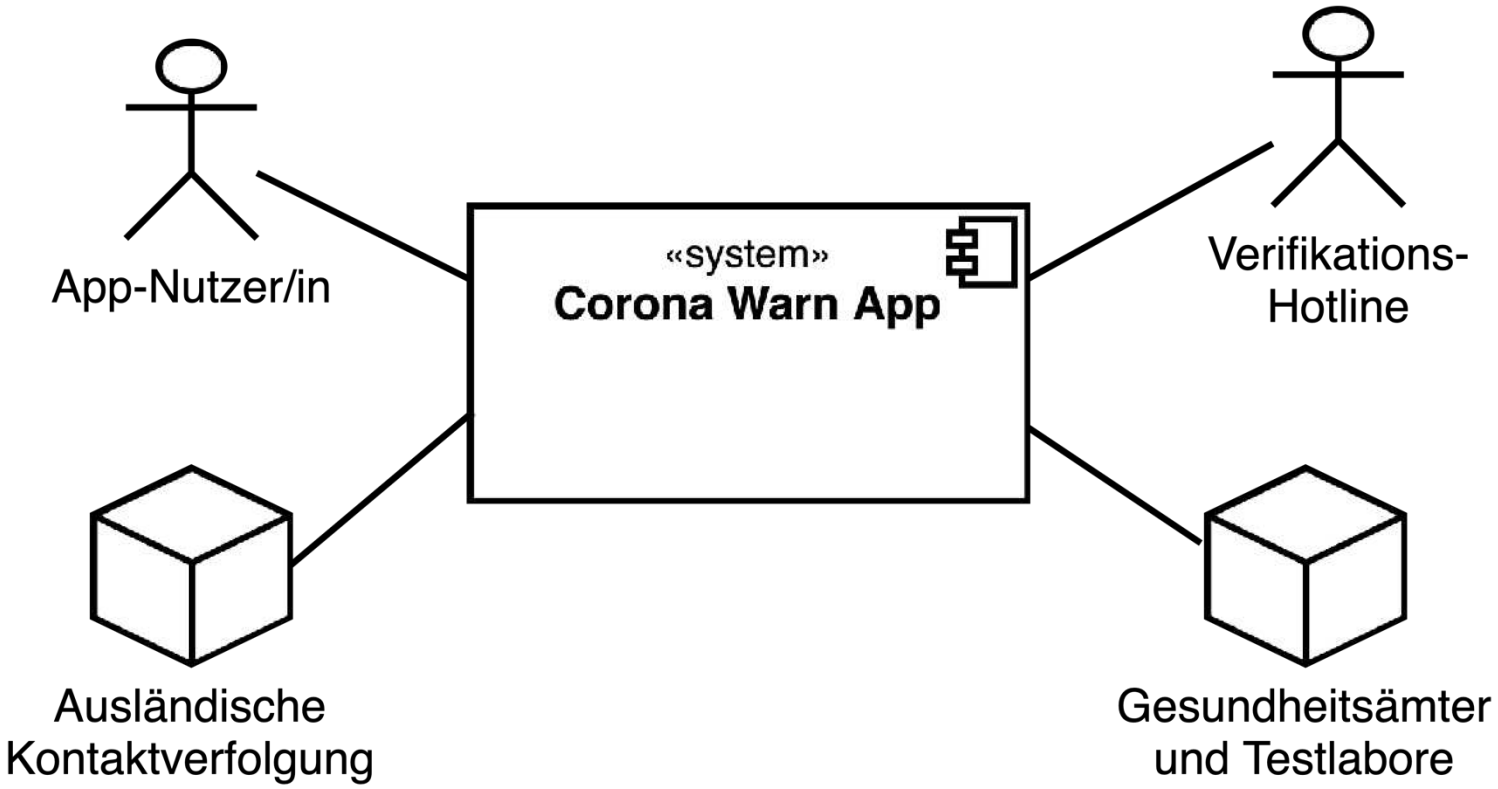
Abgrenzung des Systems und Visualisierung der Benutzer und Fremdsysteme, mit denen es interagiert.

Form: Graphik, ergänzt um kurze Beschreibungen.



Kontextabgrenzung

Dieser fachliche Systemkontext zeigt das Corona-Warn-App-System im Zusammenspiel mit den wichtigsten Benutzern und Fremdsystemen.



CORONA
WARN-APP

Legende

Benutzer
CWA-System stellt
Oberfläche bereit

Fremdsystem
CWA-System stellt
Schnittstelle(n) bereit

Interaktion

Fachlicher Systemkontext, Akteure

Kurze Erläuterungen zu den Benutzern und Fremdsystemen

Akteur	Beschreibung
App-Nutzer/in	Erhält Informationen über mögliche Begegnungen mit infizierten Personen und eigene Testergebnisse. Verifiziert eigene Testergebnisse und warnt so freiwillig andere.
Verifikations-Hotline	Unterstützt App-Nutzer/innen bei der Freischaltung positiver Testergebnisse (“teleTAN").
Gesundheitsämter und Testlabore	Liefern anonymisierte Testergebnisse an das System.
Ausländische Kontaktverfolgungen	Austausch mit dezentralen Anwendungen anderer Länder zur grenzüberschreitenden Ermittlung von Kontakten.

Was ist Softwarearchitektur?

Softwarearchitektur :=

$\sum$ wichtige Entscheidungen

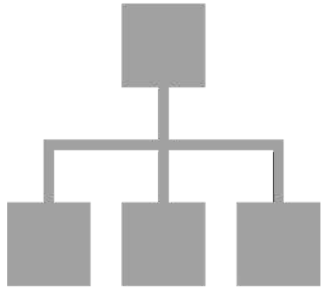
wichtig :=

- fundamental (betrifft viele)
- im weiteren Verlauf nur schwer zu ändern
- entscheidend für den Erfolg des Softwaresystems

Themen für Entscheidungen

Zerlegung

Welcher Architekturstil?
Wie zerfällt die Anwendung?
Teilsysteme, Module,
Komponentenbildung,
Abhängigkeiten ...

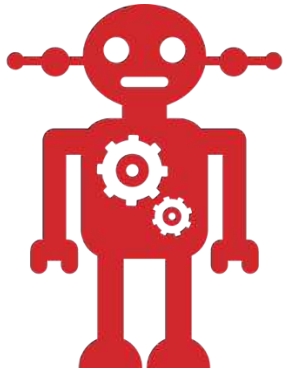


Zielumgebung

Wo läuft die Software?
Beim Endanwender, im
eigenen Rechenzentrum,
Cloud, Verteilung,
Virtualisierung ...

Technologie-Stack

Was setzen wir ein?
Programmiersprache(n)
Bibliotheken, Frameworks,
Middleware,
Querschnittsthemen ....



Vorgehen

Wie arbeiten wir?
Planen, Entwickeln, Testen,
Bauen, Dokumentieren,
Ausliefern, Nachjustieren,
...

CWA – Konkrete Entscheidungen

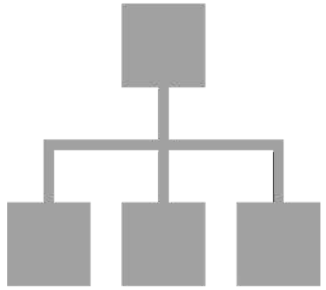
CORONA WARN-APP

embarc Software Consulting GmbH

DB

Zerlegung

Client/Server mit Apps und Backend-Server als verteilte Menge einzeln deploybarer Services, fachlich zerlegt (Test-ergebnisse, Verifikation …)

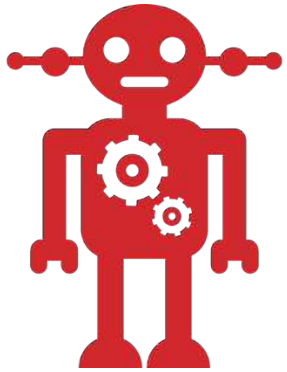


Zielumgebung

Clients: Smartphones der Endnutzer, Backend: Kubernetes in der Telekom-Cloud …

Technologie-Stack

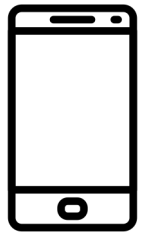
Native Apps in Swift und Kotlin auf iOS und Android.
Backend in Java mit Spring Boot, Postgres, …



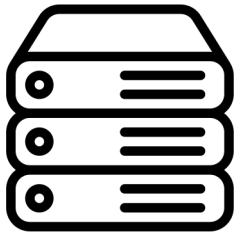
Vorgehen

Entwicklung als Open Source, Quelltexte in GitHub, Dokumentation in Markdown, automatisierte Tests
…

Technologie-Stack



- Native Clients in Swift bzw. Kotlin für iOS und Android
- SQLite



- Java 11
- Spring Boot/Cloud/Data
- Lombok, Guava, …
- REST, Protobuf
- OpenAPI, Micrometer
- Liquibase

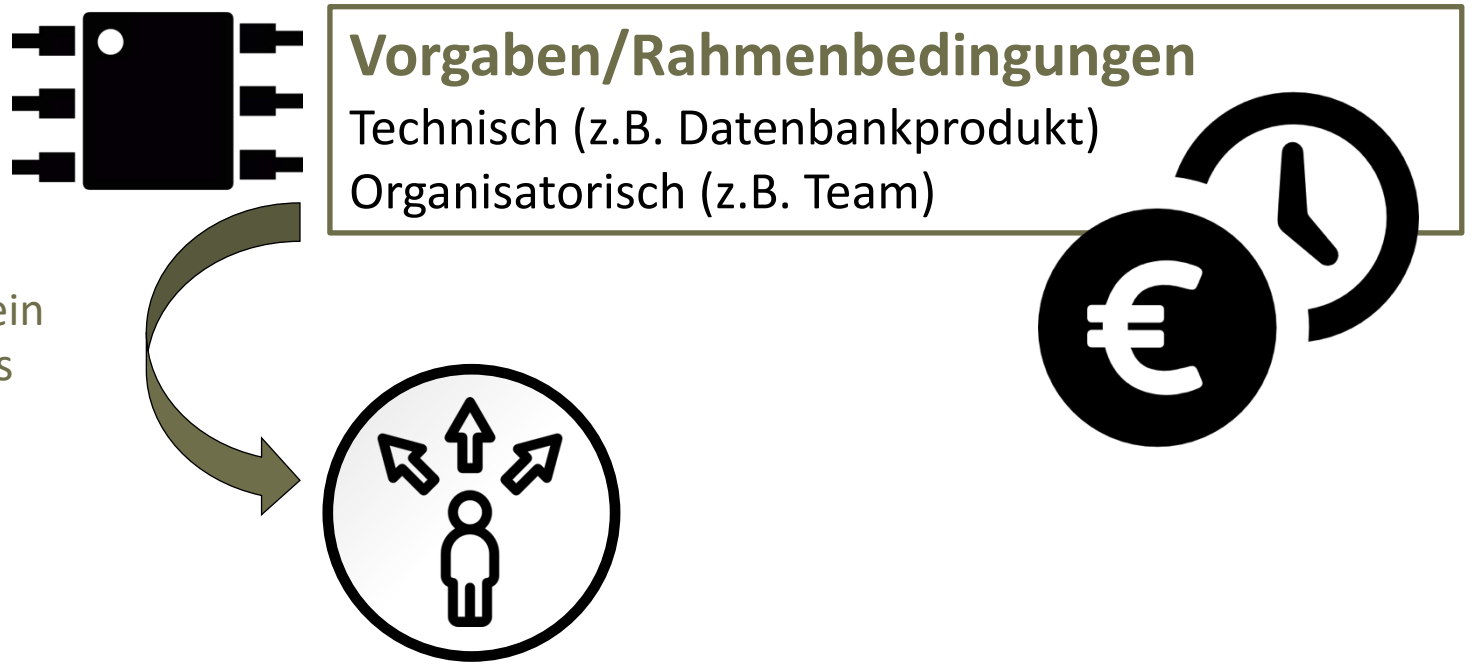
- Maven, Gradle
- Docker, Kubernetes
- Open Telekom Cloud (OpenStack)
- PostgreSQL, S3, CDN
- Keycloak

CORONA WARN-APP

embarc Software Consulting GmbH

DB

Einflüsse auf Entscheidungen



- schränken die Lösung ein
- schließen Optionen aus

Zentrale Rahmenbedingungender CWA

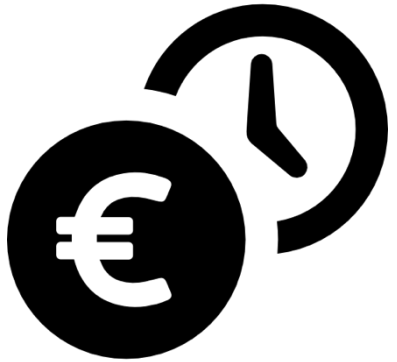
Technisch

- Betrieb in der **Cloud**
- **Native mobile** Clients
- Einsatz des **Exposure Notification Framework**

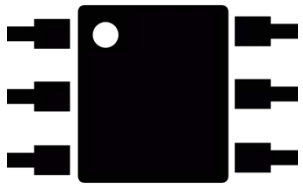
CORONA WARN-APP

Organisatorisch

- **Große Medienaufmerksamkeit**, gewisse **Skepsis** am Mehrwert innerhalb der Bevölkerung
- Konsortium aus **zwei Auftragnehmern** (SAP und Deutsche Telekom)
- **Enger Zeitrahmen**
- Hoher **politischer Druck**, viele Parteien involviert (Ministerien, Behörden, RKI)
- **Hohe Datenschutzanforderungen**



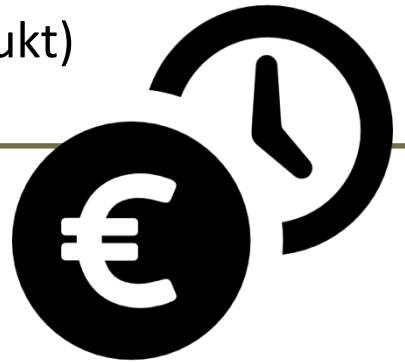
Einflüsse auf Entscheidungen



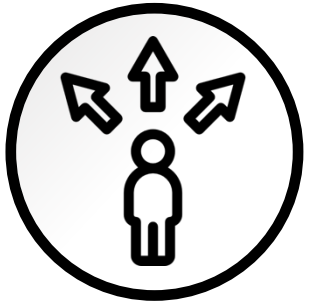
Vorgaben/Rahmenbedingungen

Technisch (z.B. Datenbankprodukt)

Organisatorisch (z.B. Team)



- schränken die Lösung ein
- schließen Optionen aus



- prägen die Lösung
- nachher schwierig “einzubauen”

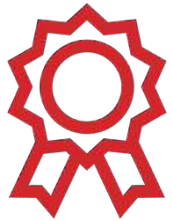
Qualitätsanforderungen

Benutzbarkeit, Effizienz,

Wartbarkeit, Sicherheit ...

Qualitätsmerkmale

Begriffe nach ISO 25010



Benutzbarkeit
(Usability)

Ist die Software intuitiv zu bedienen, leicht zu erlernen, attraktiv?

Portabilität
(Portability)

Ist die Software leicht auf andere Zielumgebungen (z.B. anderes OS) übertragbar?

Funktionale Eignung
(Functional Suitability)

Sind die berechneten Ergebnisse genau genug / exakt, ist die Funktionalität angemessen? ...

Effizienz
(Performance)

Antwortet die Software schnell, hat sie einen hohen Durchsatz, einen geringen Ressourcenverbrauch? ...

Kompatibilität
(Compatibility)

Ist die Software konform zu Standards, arbeitet sie gut mit anderen zusammen?

Zuverlässigkeit
(Reliability)

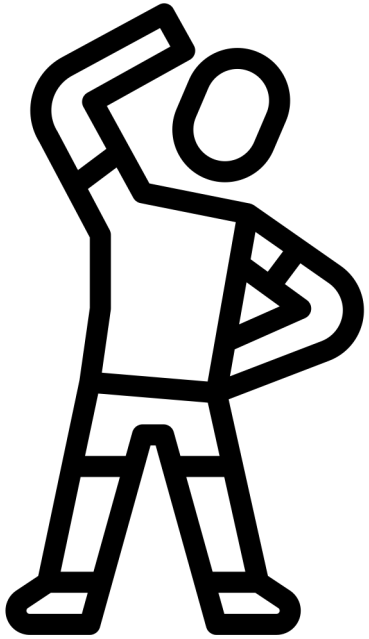
Ist das System verfügbar, tolerant gegenüber Fehlern, nach Abstürzen schnell wieder hergestellt? ...

Sicherheit
(Security)

Ist das System sicher vor Angriffen? Sind Daten und Funktion vor unberechtigtem Zugriff geschützt? ...

Wartbarkeit
(Maintainability)

Ist die Software leicht zu ändern, erweitern, testen, verstehen? Lassen sich Teile wiederverwenden? ...



Bitte folgt dem Link für eine kleine Übung (04)

https://tinyurl.com/etffm-docs

Top-Qualitätsziele Corona-Warn-App

	Ziel	Beschreibung
	**Höchster Datenschutz**	Der Schutz der personenbezogenen Daten hat oberste Priorität. *(Sicherheit)*
	**Effektive Warnfunktionalität**	Die App ist ein effektiver Baustein bei der Pandemie-Bekämpfung. *(Funktionale Eignung)*
	**Attraktive Lösung für App-Nutzer**	Die App ist leicht zu installieren sowie intuitiv und effizient zu bedienen. *(Benutzbarkeit)*
	**Hohe Zuverlässigkeit**	Die Lösung geht mit Lastspitzen wegen hoher Nutzer- oder Infektionszahlen ebenso souverän um, wie mit böswilligen Angriffen. *(Zuverlässigkeit)*
	**Gute Änderbarkeit**	Die Software lässt sich leicht anpassen, wenn z. B. Nutzer/-innen oder neue Forschungsergebnisse es erfordern. *(Wartbarkeit/Erweiterbarkeit)*

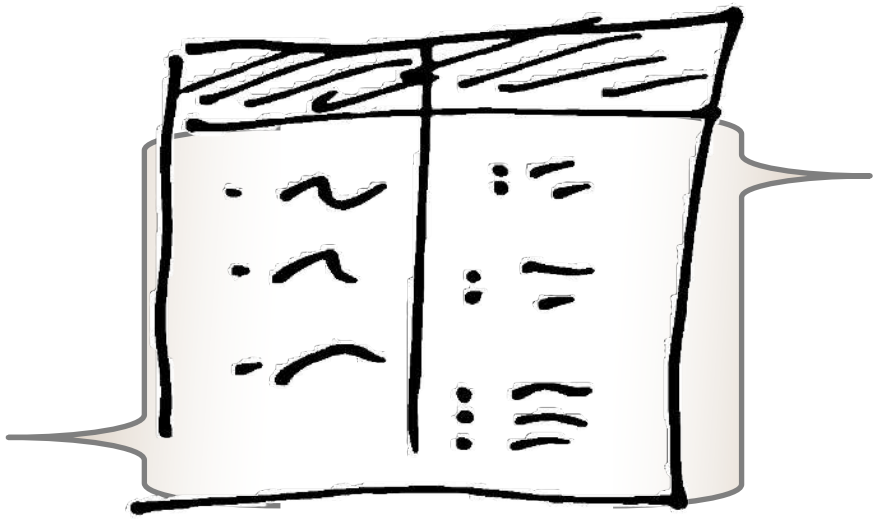
Die Reihenfolge gibt Orientierung bezüglich der Wichtigkeit.

Lösungsstrategie

Stellt Qualitätsziele und zugeordnete high-level Lösungsansätze in Beziehung zueinander dar.

Form: Tabelle ([Ziele | Lösungsansätze]).

Architektur-
/Qualitätsziele



Architektur-
/Lösungsansätze

Lösungsstrategie Corona-Warn-App

Ziel	Passende Lösungsansätze (Auswahl)
 **Höchster Datenschutz**	• Speicherung der **Daten lokal** • **Verschlüsselung** aller Bewegungsdaten • Senden der Daten nur **nach Aufforderung** • **Transparente** Entwicklung (Open Source)
 **Effektive Warnfunktionalität**	• Verwendung **Exposure Notification Framework** • **Digitale Abläufe** bevorzugt • Optionales, manuelles **Kontakt-Tagebuch**
 **Attraktive Lösung für App-Nutzer**	• **Native Clients** (L&F) • **Übersichtliche** Gestaltung und **simple** Bedienung
 **Hohe Zuverlässigkeit**	• **Microservices** • Docker, Kubernetes, **Public Cloud** • Bereitstellung von zu lesenden Daten über **CDN**
 **Gute Änderbarkeit**	• Hoher **Modularisierungsgrad** • **Open Source** Projekt**,** gute **Dokumentation** • Verwendung von **Standard** & **Open Source** Libraries • Konsortium von **mehreren Auftragnehmern** • **Code-Qualität** (SonarQube, SwiftLint, Checkstyle, …)

CORONA WARN-APP

WAS sollen wir bauen?

Was wollen wir erreichen?
Wozu ist es da?
Wem nützt es?
Was soll es tun?
Was braucht es nicht tun?
Woran halten wir uns?
Was soll es exzellent können?

ANFORDERUNGEN

Architekturüberblick

Aufgabe

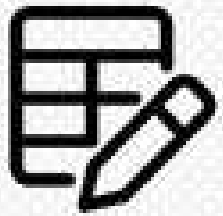
- Mission Statement
- Kontextabgrenzung

Einflüsse

- Rahmenbedingungen
- Qualitätsziele

Lösungsansätze

- Architekturstil
- Technologie-Stack
- Konzepte
- Prinzipien
- Zerlegung
- ....



Lösungsstrategie (plus Überblicksbild) als Brücke

WIE sieht die Lösung aus?

Wie erreichen wir das?
Welchen Mustern folgen wir?
Was verwenden wir?
Was leitet uns?
Woraus besteht es?
Wie ist es strukturiert?

ENTSCHEIDUNGEN

embarc.de

Softwarearchitektur auf dem Bierdeckel

„Abheften“ in arc42

1. Einführung und Ziele
1.1 Aufgabenstellung
1.2 Qualitätsziele
1.3 Stakeholder

2. Randbedingungen
2.1 Technische Randbedingungen
2.2 Organisatorische Randbedingungen
2.3 Konventionen

3. Kontextabgrenzung
3.1 Fachlicher Kontext
3.2 Technischer- oder Verteilungskontext

4. Lösungsstrategie

5. Bausteinsicht
5.1 Ebene 1
5.2 Ebene 2
...

6. Laufzeitsicht
6.1 Laufzeitszenario 1
6.2 Laufzeitszenario 2
...

7. Verteilungssicht
7.1 Infrastruktur Ebene 1
7.2 Infrastruktur Ebene 2
...

8. Konzepte
8.1 Fachliche Strukturen und Modelle
8.2 Typische Muster und Strukturen
8.3 Persistenz
8.4 Benutzungsoberfläche
...

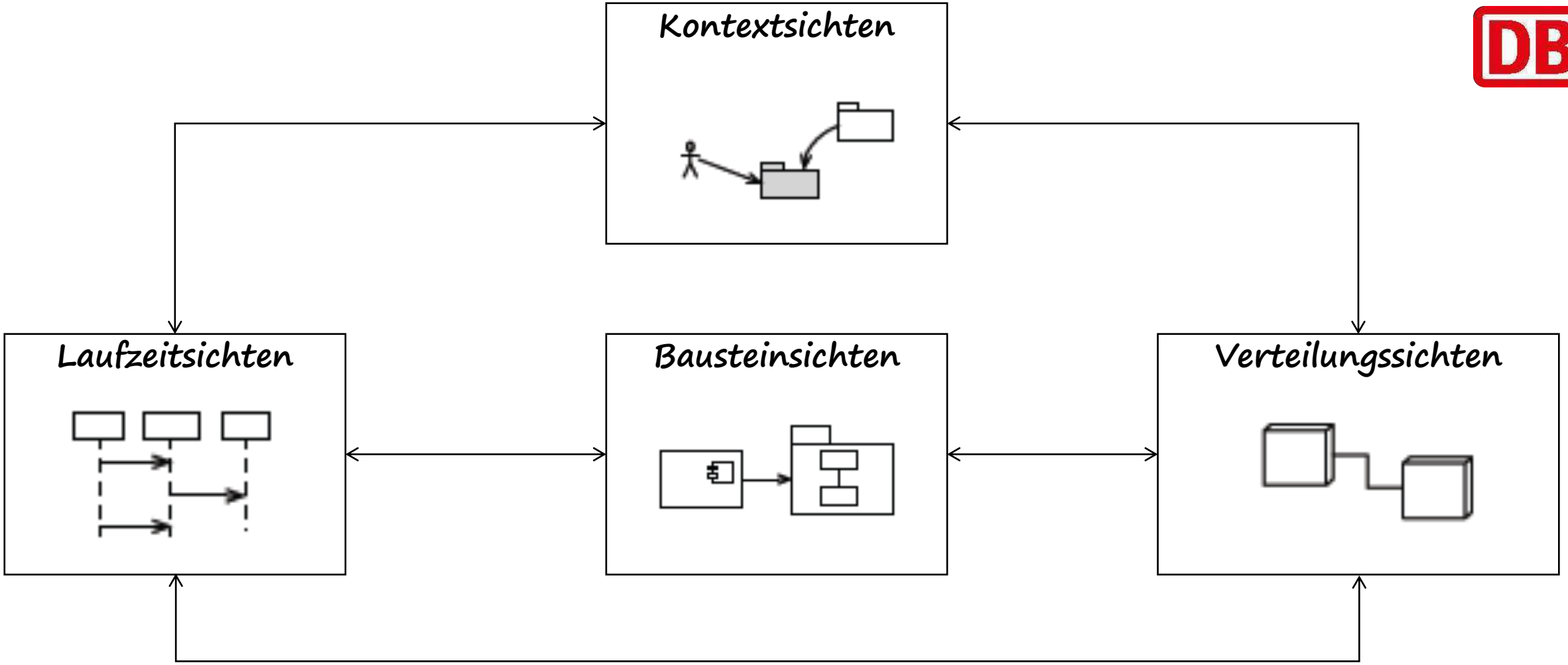
9. Entwurfsentscheidungen
9.1 Entwurfsentscheidung 1
9.2 Entwurfsentscheidung 2
...

10. Qualitätsszenarien
10.1 Qualitätsbaum
10.2 Bewertungsszenarien

11. Risiken

12. Glossar

➔https://arc42.de/


Kontextsichten
Laufzeitsichten
Bausteinsichten
Verteilungssichten

Sichten in arc42

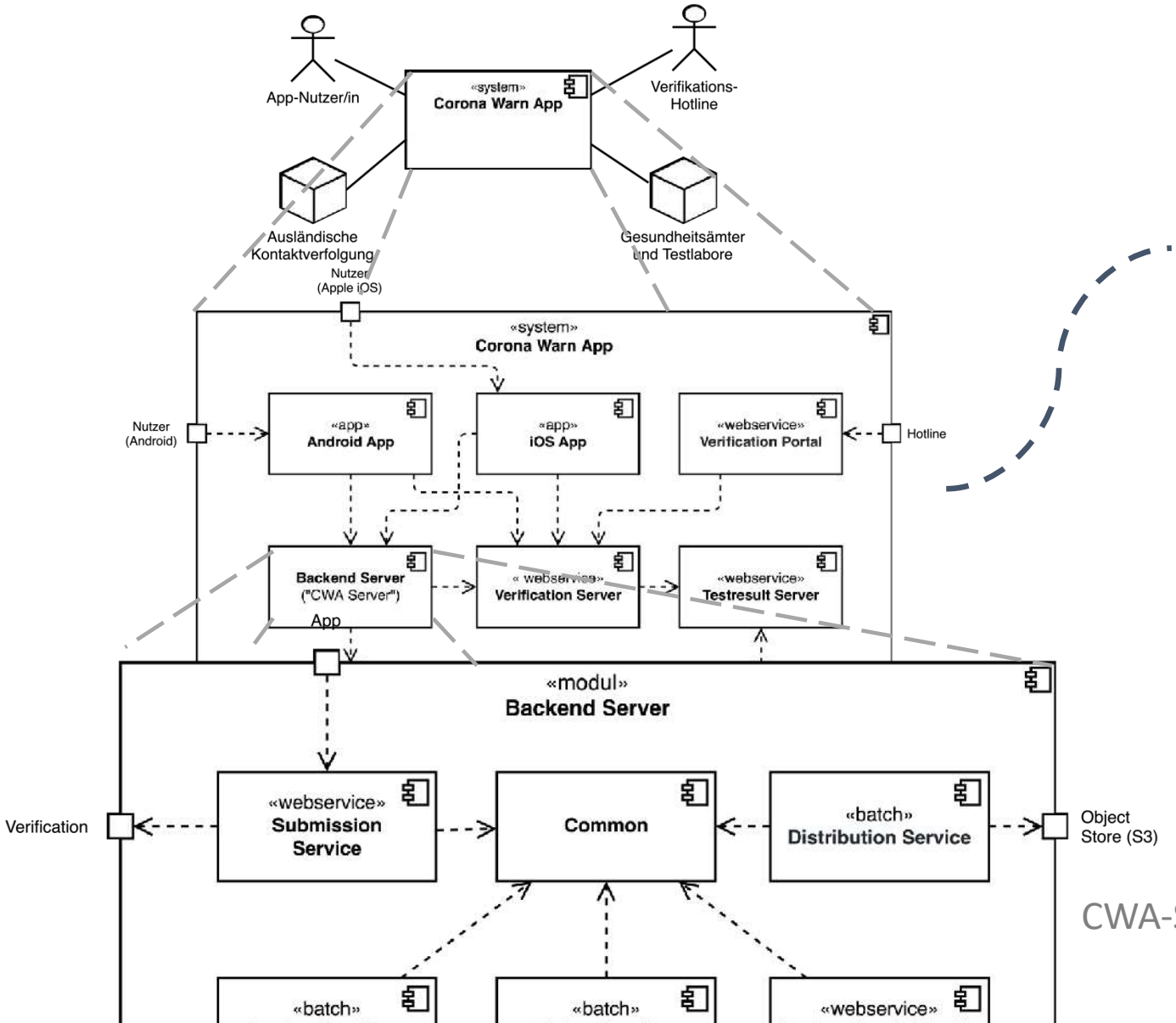
Kontextabgrenzung (System Context)
Zeigt die beteiligten Fremdsysteme und Benutzer, mit denen das zu beschreibende System interagiert.

Bausteinsicht (Building Block View)
Zeigt die Struktur des Softwaresystems in Form seiner Bausteine, und wie diese zusammenhängen. Bündelt viele Entscheidungen rund um Verantwortlichkeiten.

Laufzeitsicht (Runtime View)
Zeigt Elemente der Baustein- und/oder Kontextsicht in Aktion, veranschaulicht dynamische Strukturen und Verhalten des beschriebenen Systems.

Verteilungssicht (Deployment View)
Zeigt wie die Software in Betrieb genommen (verteilt) wird, und wie die Zielumgebung aussieht.

Tatsächliche Zerlegung im Quelltext



„Bausteinsicht, Ebene 1“

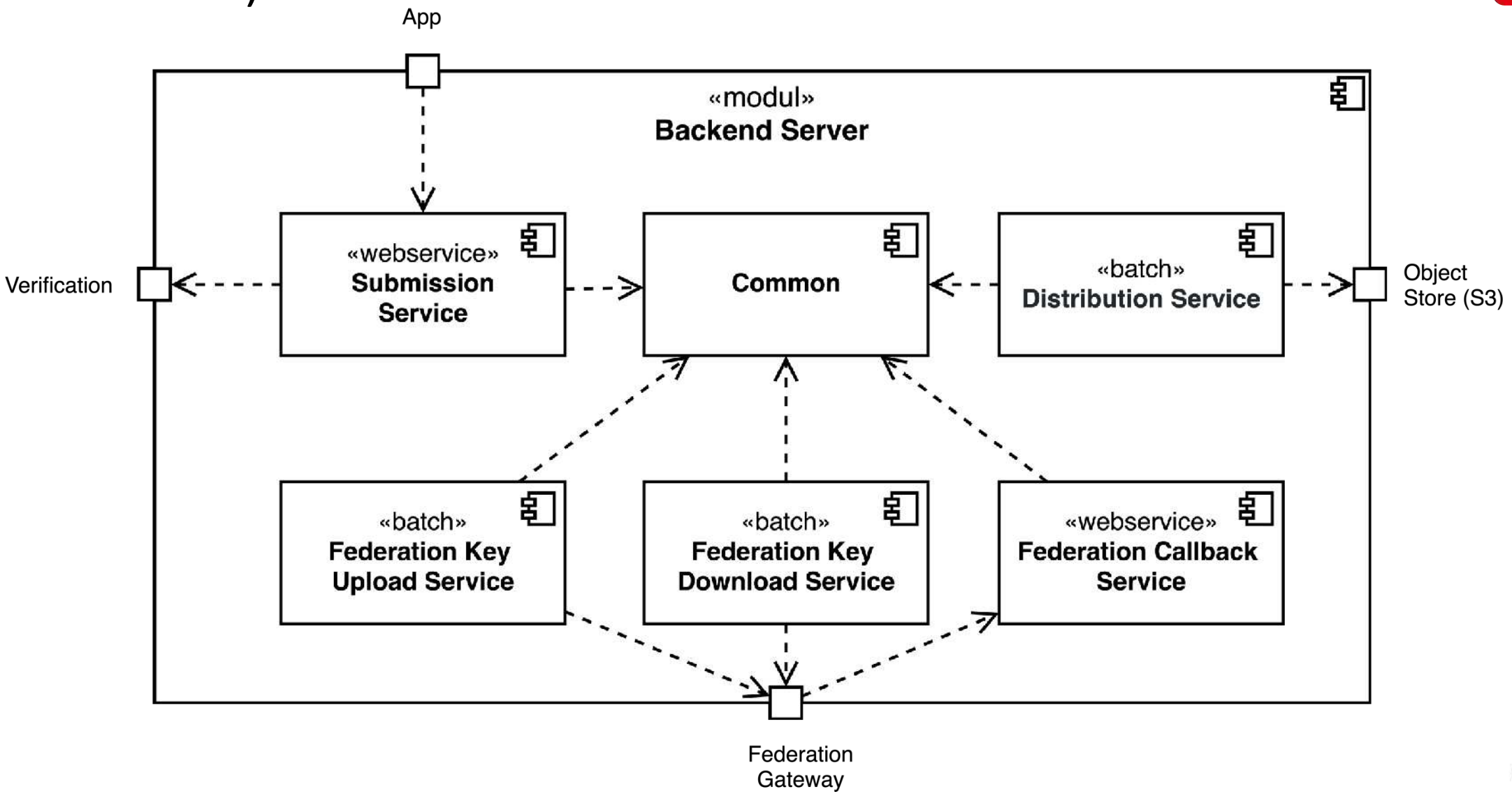
GitHub-Repositories

https://github.com/corona-warn-app/...

- cwa-app-android
- cwa-app-ios
- cwa-server („Backend“)
- cwa-testresult-server
- cwa-verification-server
- cwa-verification-portal

CWA-Server (Backend), Bausteinsicht, Ebene 2

Zerlegung Backend Server (Bausteinsicht, Ebene 2)



„Abheften“ in arc42

➔https://arc42.de/

Cross-cutting concerns

Ziele

- Einheitliche Lösungen, Konsistenz
- weniger Redundanzen
- Bessere Wartbarkeit

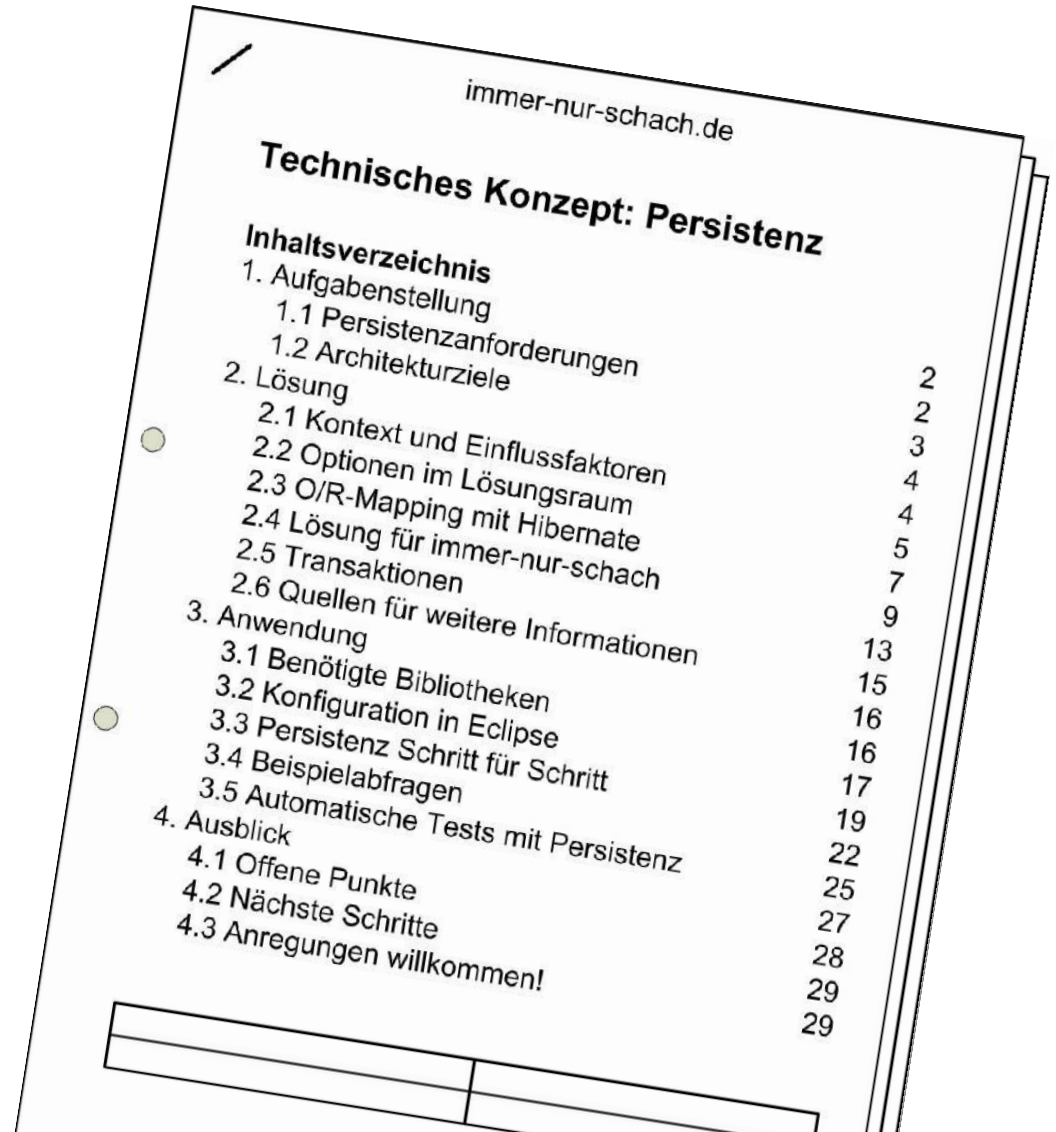
Lösungsraum

- (Architektur-)muster
- Bibliotheken, Frameworks
- Aspektorientierte Programmierung
- Generierung
- …

Vorschlag Hauptüberschriften

(pro Konzept)

1. Aufgabenstellung
2. Lösung
3. Anwendung
4. Ausblick



immer-nur-schach.de

Technisches Konzept: Persistenz

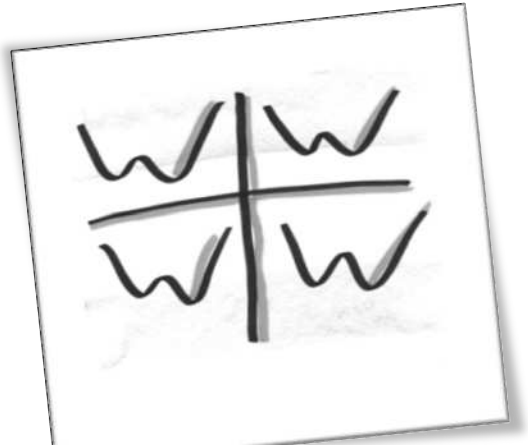
Inhaltsverzeichnis

1. Aufgabenstellung 2
   1.1 Persistenzanforderungen 2
   1.2 Architekturziele 3
2. Lösung 4
   2.1 Kontext und Einflussfaktoren 4
   2.2 Optionen im Lösungsraum 5
   2.3 O/R-Mapping mit Hibernate 7
   2.4 Lösung für immer-nur-schach 9
   2.5 Transaktionen 13
   2.6 Quellen für weitere Informationen 15
3. Anwendung 16
   3.1 Benötigte Bibliotheken 16
   3.2 Konfiguration in Eclipse 17
   3.3 Persistenz Schritt für Schritt 19
   3.4 Beispielabfragen 22
   3.5 Automatische Tests mit Persistenz 25
4. Ausblick 27
   4.1 Offene Punkte 28
   4.2 Nächste Schritte 29
   4.3 Anregungen willkommen! 29

Zutat “Konzept”

Gleiche Dinge auch gleich lösen. Zentrale, übergreifende Themen einmal bearbeiten, bewerten und dann geeignet festhalten und kommunizieren.

- Das 4-Quadrantenmodell hilft beim Strukturieren von Konzepten.



„Abheften“ in arc42

1. Einführung und Ziele
1.1 Aufgabenstellung
1.2 Qualitätsziele
1.3 Stakeholder

2. Randbedingungen
2.1 Technische Randbedingungen
2.2 Organisatorische Randbedingungen
2.3 Konventionen

3. Kontextabgrenzung
3.1 Fachlicher Kontext
3.2 Technischer- oder Verteilungskontext

4. Lösungsstrategie

5. Bausteinsicht
5.1 Ebene 1
5.2 Ebene 2
...

6. Laufzeitsicht
6.1 Laufzeitszenario 1
6.2 Laufzeitszenario 2
...

7. Verteilungssicht
7.1 Infrastruktur Ebene 1
7.2 Infrastruktur Ebene 2
...

8. Konzepte
8.1 Fachliche Strukturen und Modelle
8.2 Typische Muster und Strukturen
8.3 Persistenz
8.4 Benutzungsoberfläche
...

9. Entwurfsentscheidungen
9.1 Entwurfsentscheidung 1
9.2 Entwurfsentscheidung 2
...

10. Qualitätsszenarien
10.1 Qualitätsbaum
10.2 Bewertungsszenarien

11. Risiken

12. Glossar

➔https://arc42.de/

Softwarearchitektur

Eine der Definitionen …

> *“Softwarearchitektur ist die **Menge der Entwurfsentscheidungen**, die, **wenn falsch** getroffen, Dein Projekt zum **Scheitern bringen kann**.”**

(Eoin Woods)

* Im Englischen eigentlich: “Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled.”

7 Regeln für gute Dokumentation

„Documenting Software Architectures: Views and Beyond“
Clements, et.al, 2. Auflage 2010

1. Schreibe aus Sicht des Lesers
2. Vermeide unnötige Wiederholungen
3. Vermeide Mehrdeutigkeiten
    3. a) Erkläre Deine Notation
4. Verwende eine Standardstrukturierung
5. **Halte Begründungen für Entscheidungen fest**
6. Halte Dokumentation aktuell, aber auch nicht zu aktuell
7. Überprüfe Dokumentation auf ihre Gebrauchstauglichkeit

Zutat “Architekturentscheidung”

Zentrale Entscheidungen nachvollziehbar festhalten ist der Schlüssel, um bessere Antworten zu haben als „Historisch gewachsen“.

- Sich an der Entscheidungsmindmap entlang zu hangeln gibt Orientierung.

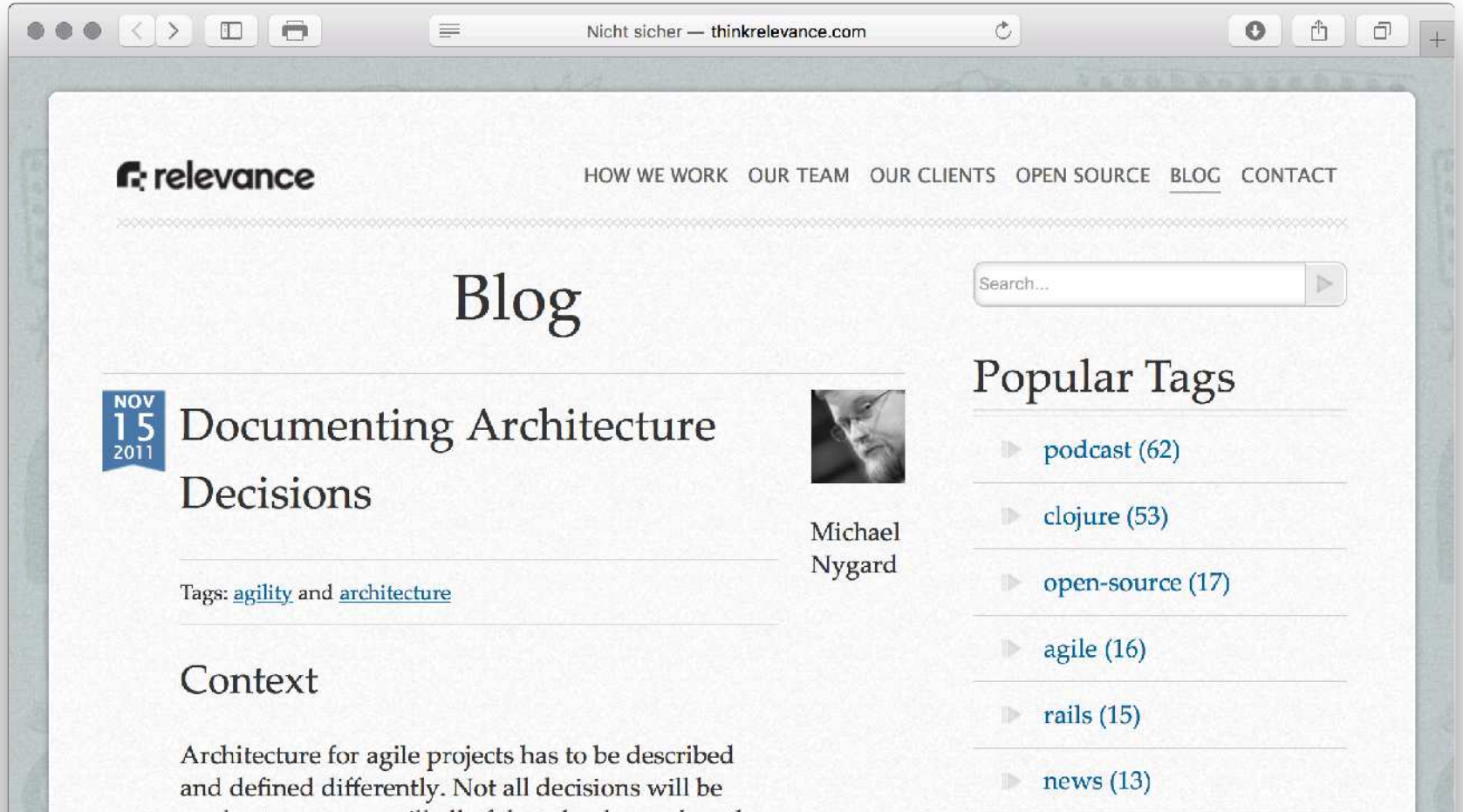
Ein Werkzeug

Zum Entscheidungen treffen und festhalten ...

Alternative: ADRs

Architecture Decision Records (Michael Nygard, 2011)

➔ http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions



„Abheften“ in arc42

1. Einführung und Ziele
1.1 Aufgabenstellung
1.2 Qualitätsziele
1.3 Stakeholder

2. Randbedingungen
2.1 Technische Randbedingungen
2.2 Organisatorische Randbedingungen
2.3 Konventionen

3. Kontextabgrenzung
3.1 Fachlicher Kontext
3.2 Technischer- oder Verteilungskontext

4. Lösungsstrategie

5. Bausteinsicht
5.1 Ebene 1
5.2 Ebene 2
...

6. Laufzeitsicht
6.1 Laufzeitszenario 1
6.2 Laufzeitszenario 2
...

7. Verteilungssicht
7.1 Infrastruktur Ebene 1
7.2 Infrastruktur Ebene 2
...

8. Konzepte
8.1 Fachliche Strukturen und Modelle
8.2 Typische Muster und Strukturen
8.3 Persistenz
8.4 Benutzungsoberfläche
...

9. Entwurfsentscheidungen
9.1 Entwurfsentscheidung 1
9.2 Entwurfsentscheidung 2
...

10. Qualitätsszenarien
10.1 Qualitätsbaum
10.2 Bewertungsszenarien

11. Risiken

12. Glossar

➔https://arc42.de/

Was ist ein Szenario?

Ein Qualitätsszenario (auch: Bewertungsszenario) ...

- ... ist ein kurzer Text (1-3 Sätze).
- ... beschreibt **beispielhaft** die Verwendung des Systems, und zwar so dass ein **Qualitätsmerkmal** die Hauptrolle spielt.

Wie konkret?

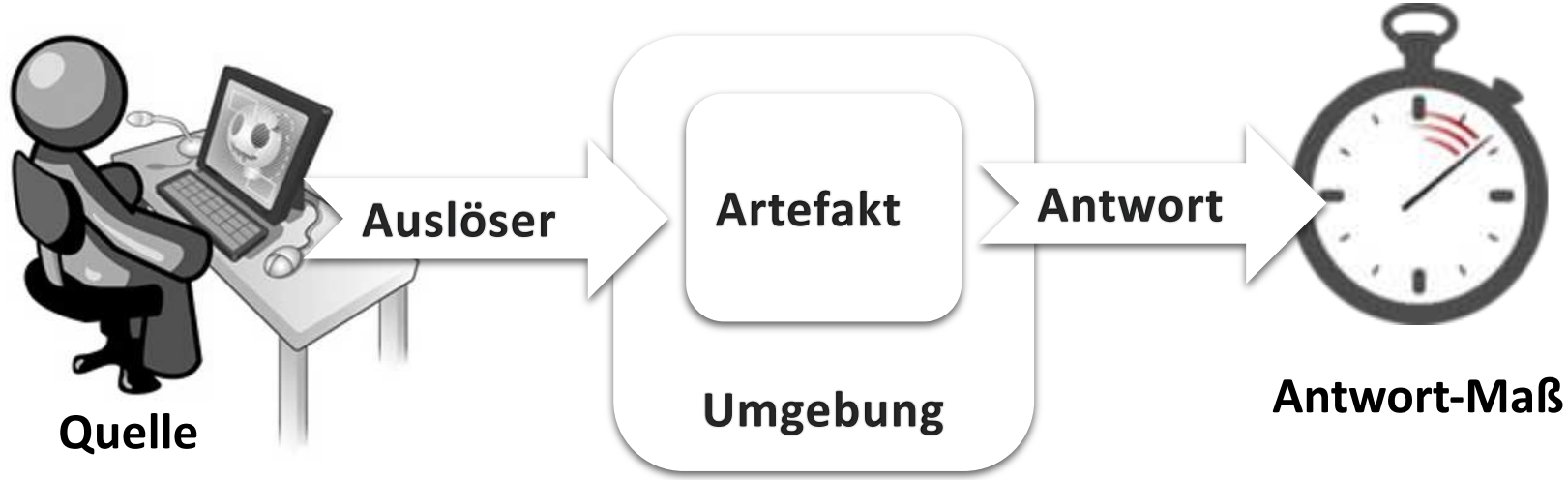
Man muss sinnvoll drüber reden können.

Man muss es (theoretisch) überprüfen können.

(Kein Abnahmekriterium, kein Testfall!)

Bestandteile eines Szenarios

Typische Bestandteile eines Qualitätsszenarios:



> *Ein angemeldeter Schach-Polizist ruft über das Internet den Bericht zu verdächtigen Spielern auf und erhält das Ergebnis innerhalb von 5 Sekunden.*

Szenarienbasierte Bewertung

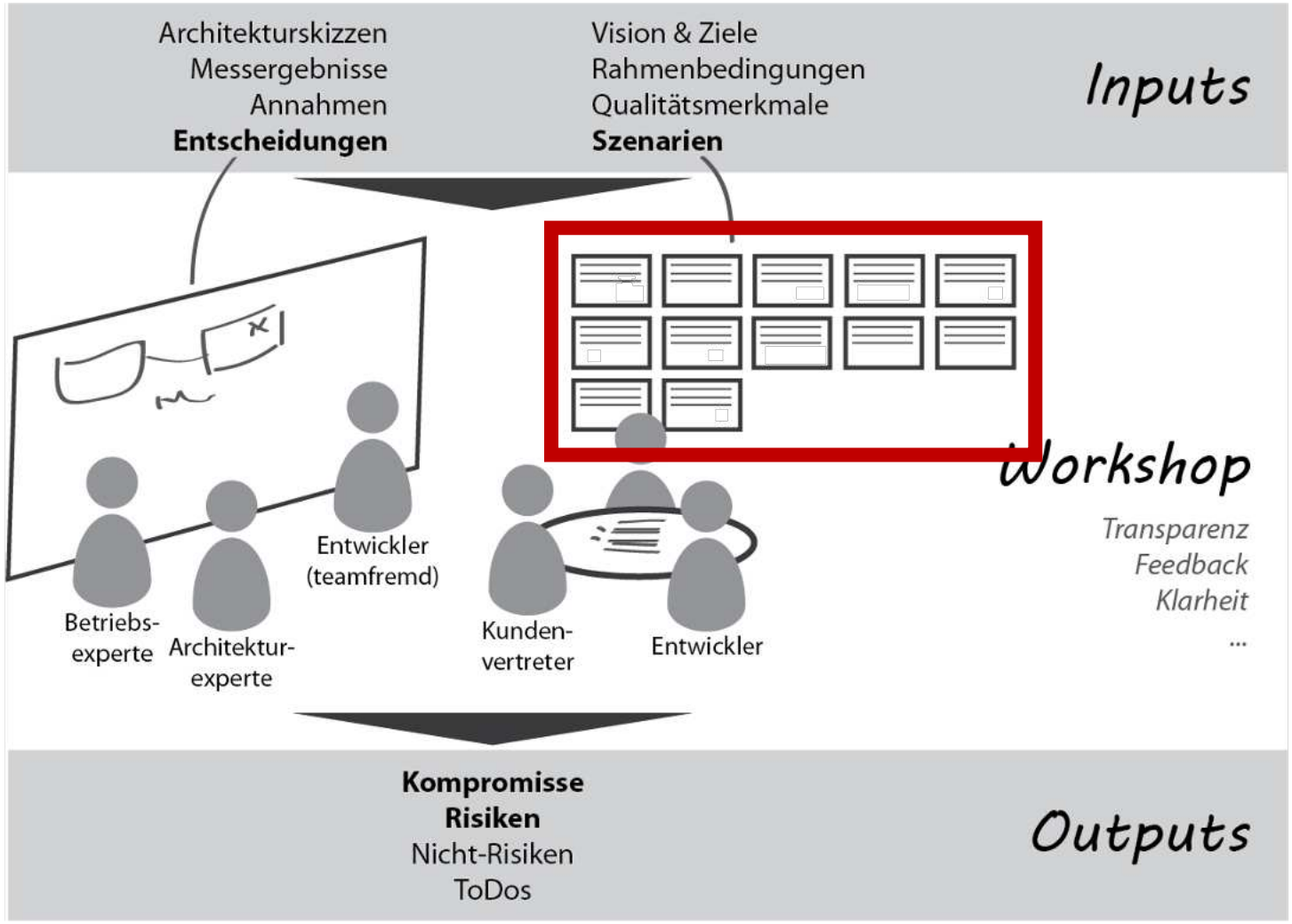
Ablauf

(1) Szenarien generieren

Mögliche Techniken:
- Brainstorming
- Ableiten aus Dokumenten
- Interviews mit Stakeholdern

(2) Szenarien priorisieren

(3) Szenarien durchsprechen
(der Reihe, nach Priorität)



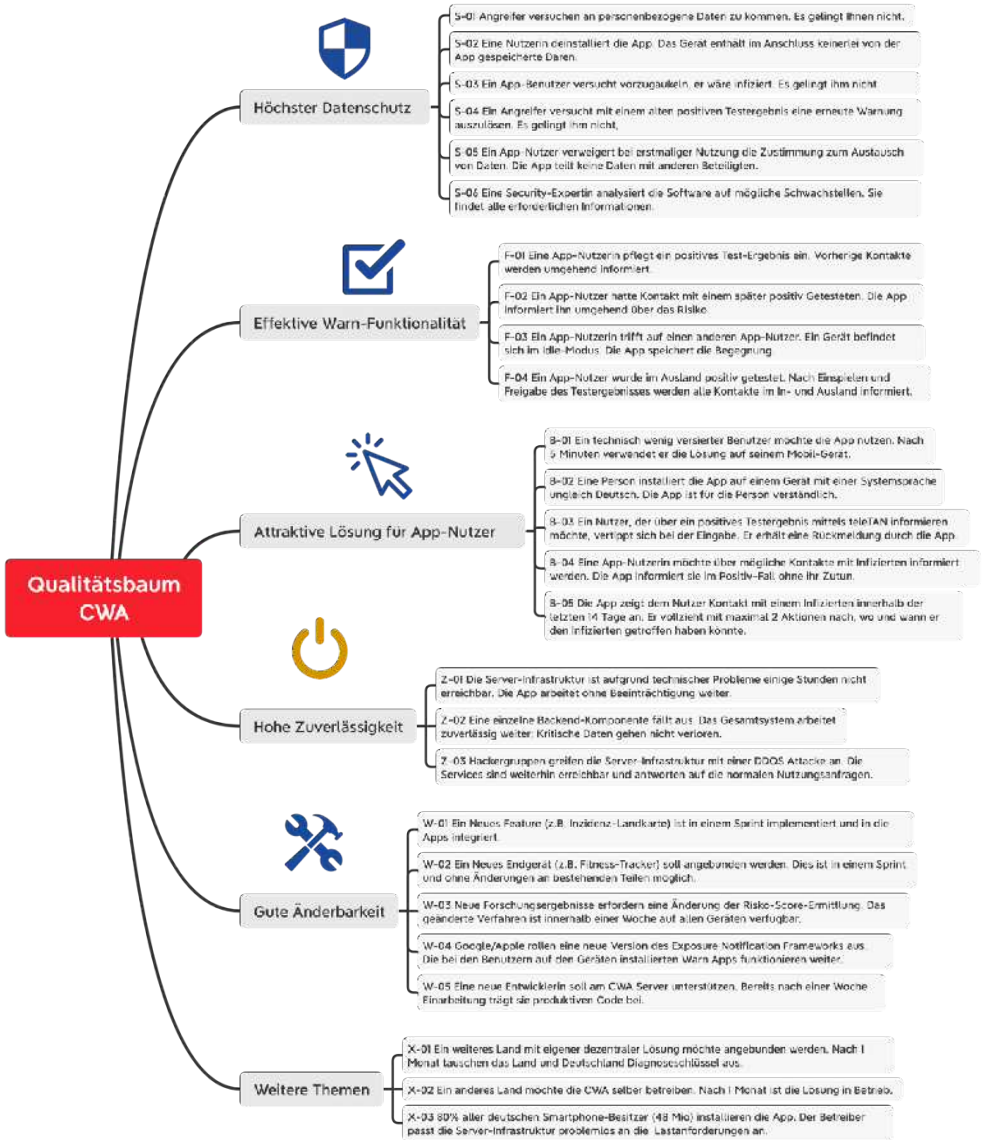
Quelle der Abbildung: S. Toth, Vorgehensmuster für Softwarearchitektur, 3. Auflage, Hanser 2019

Brainstorming – analog oder digital

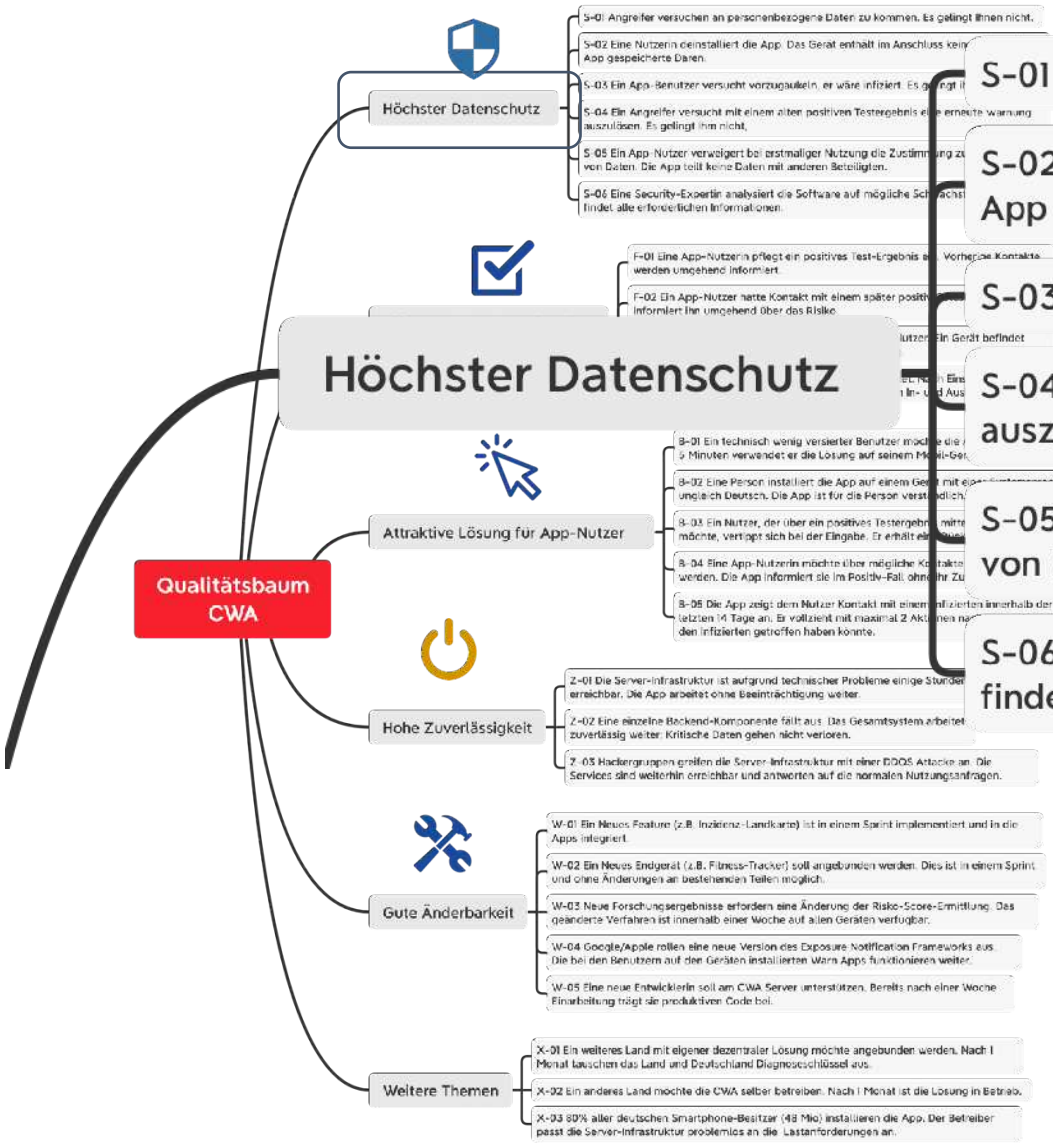
In Präsenz:
Post-its

Remote:
Digitale Whiteboards
(miro, Mural, …)

Ergebnis: Qualitätsbaum mit Szenarien



Ergebnis: Qualitätsbaum mit Szenarien



S-01 Angreifer versuchen an personenbezogene Daten zu kommen. Es gelingt Ihnen nicht.

S-02 Eine Nutzerin deinstalliert die App. Das Gerät enthält im Anschluss keinerlei von der App gespeicherte Daren.

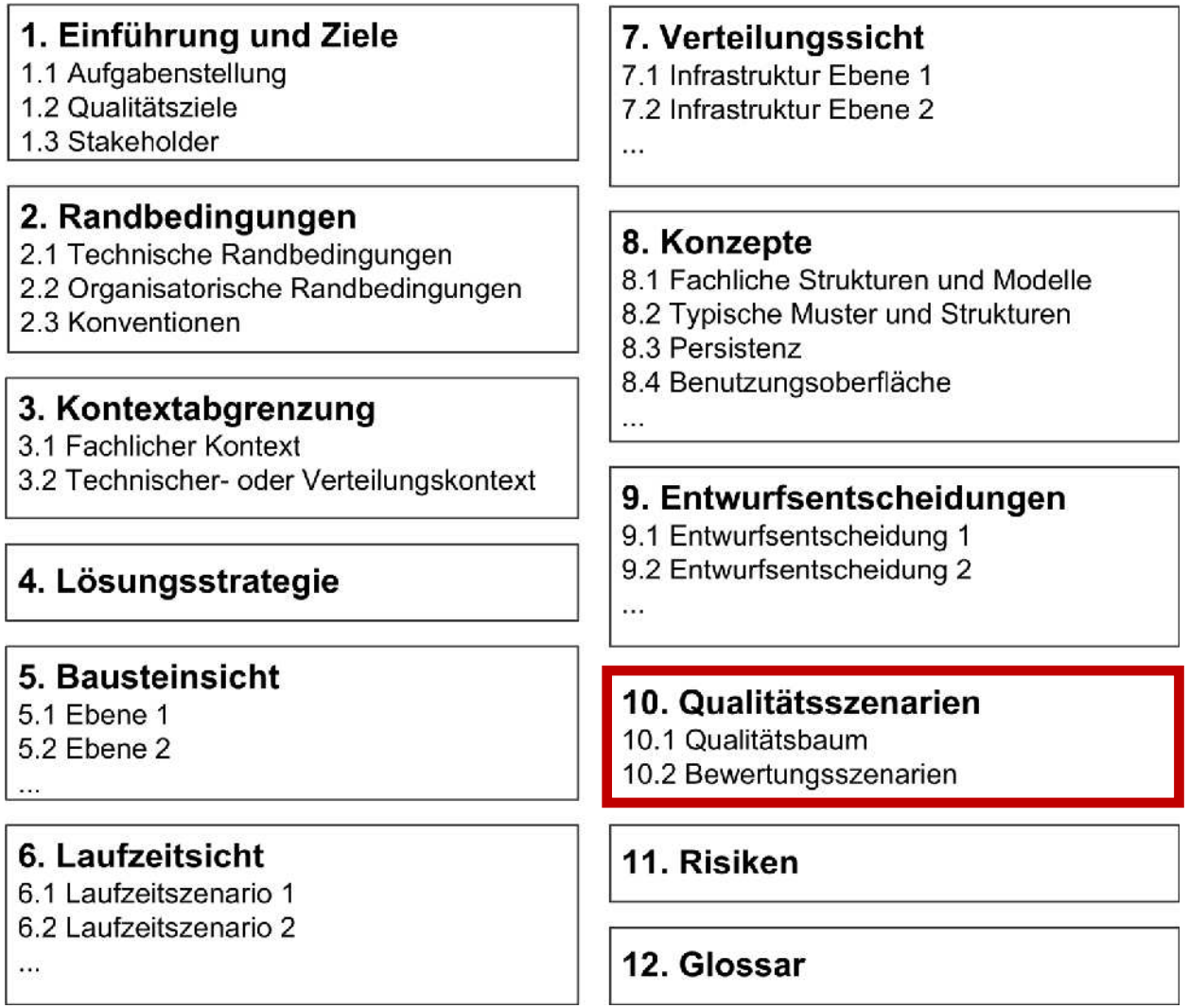
S-03 Ein App-Benutzer versucht vorzugaukeln, er wäre infiziert. Es gelingt ihm nicht.

S-04 Ein Angreifer versucht mit einem alten positiven Testergebnis eine erneute Warnung auszulösen. Es gelingt ihm nicht,

S-05 Ein App-Nutzer verweigert bei erstmaliger Nutzung die Zustimmung zum Austausch von Daten. Die App teilt keine Daten mit anderen Beteiligten.

S-06 Eine Security-Expertin analysiert die Software auf mögliche Schwachstellen. Sie findet alle erforderlichen Informationen.

„Abheften“ in arc42



➔https://arc42.de/

Kurze Pause bis 10:45 Uhr

Agenda

- Einführung
- Documentation as Code
- docToolchain
- Architektur dokumentieren
- **Praktische Umsetzung**
- Weiterführende Themen
- Fazit und Ausblick

```
cp -r ../resources/* .
```

./src/docs/AsciiDocBasics.adoc

* AsciiDoc Formatierungen
* Listen
* Attributes
* Bilder
* Tabellen
* Anchors
* Source-Code
* Modulare Dokumentation

Modulare Dokumentation

arc42-master.adoc

include

kapitel1.adoc

include

business-master.adoc

kapitel2.adoc

include

```
public class HelloWorld {
   public static void main (String[] args) {
     // Ausgabe Hello World!
     System.out.println("Hello World!");
   }
}
```

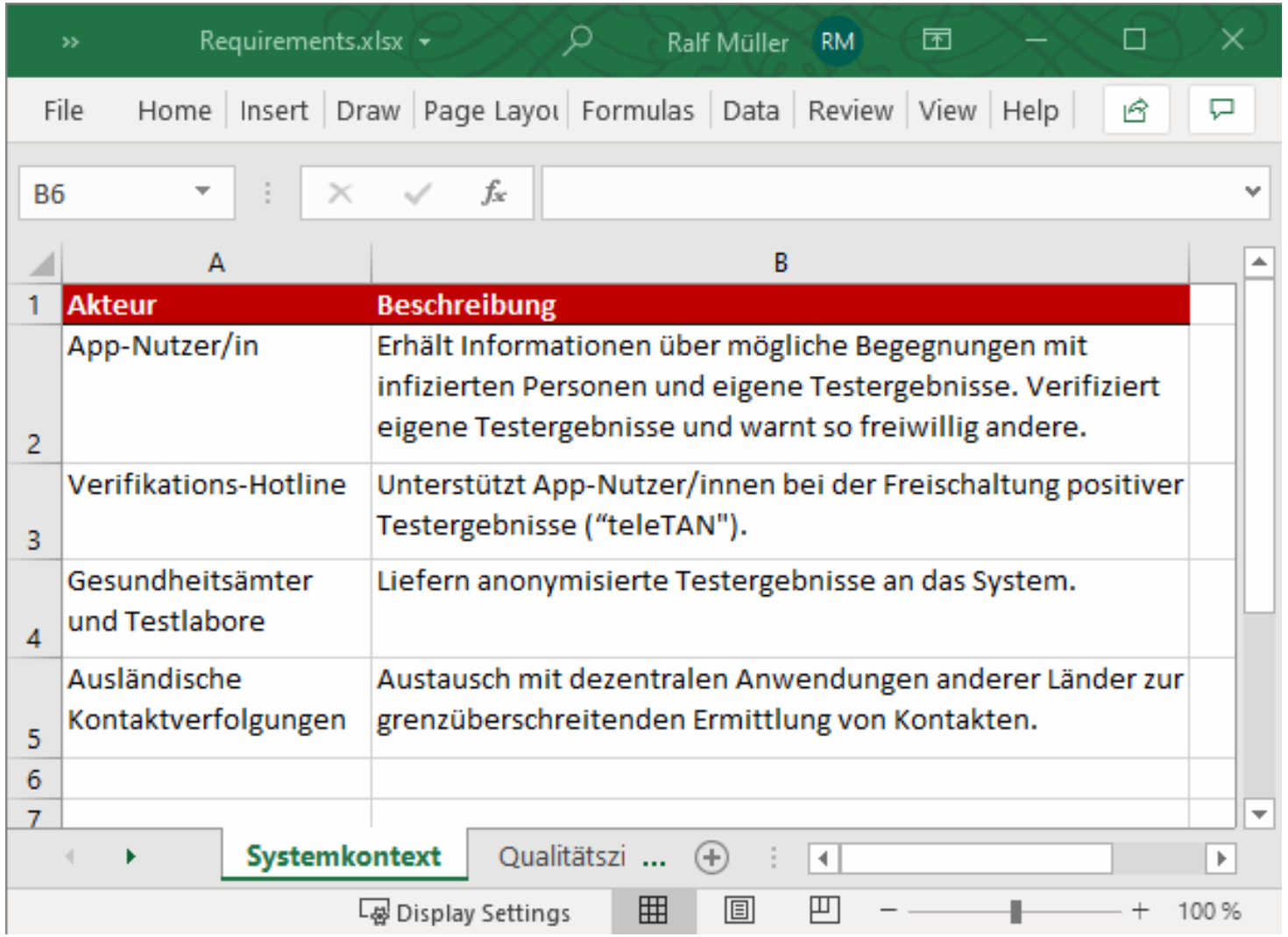
security-master.adoc

kapitel8.adoc

include

kapitel8.1.adoc

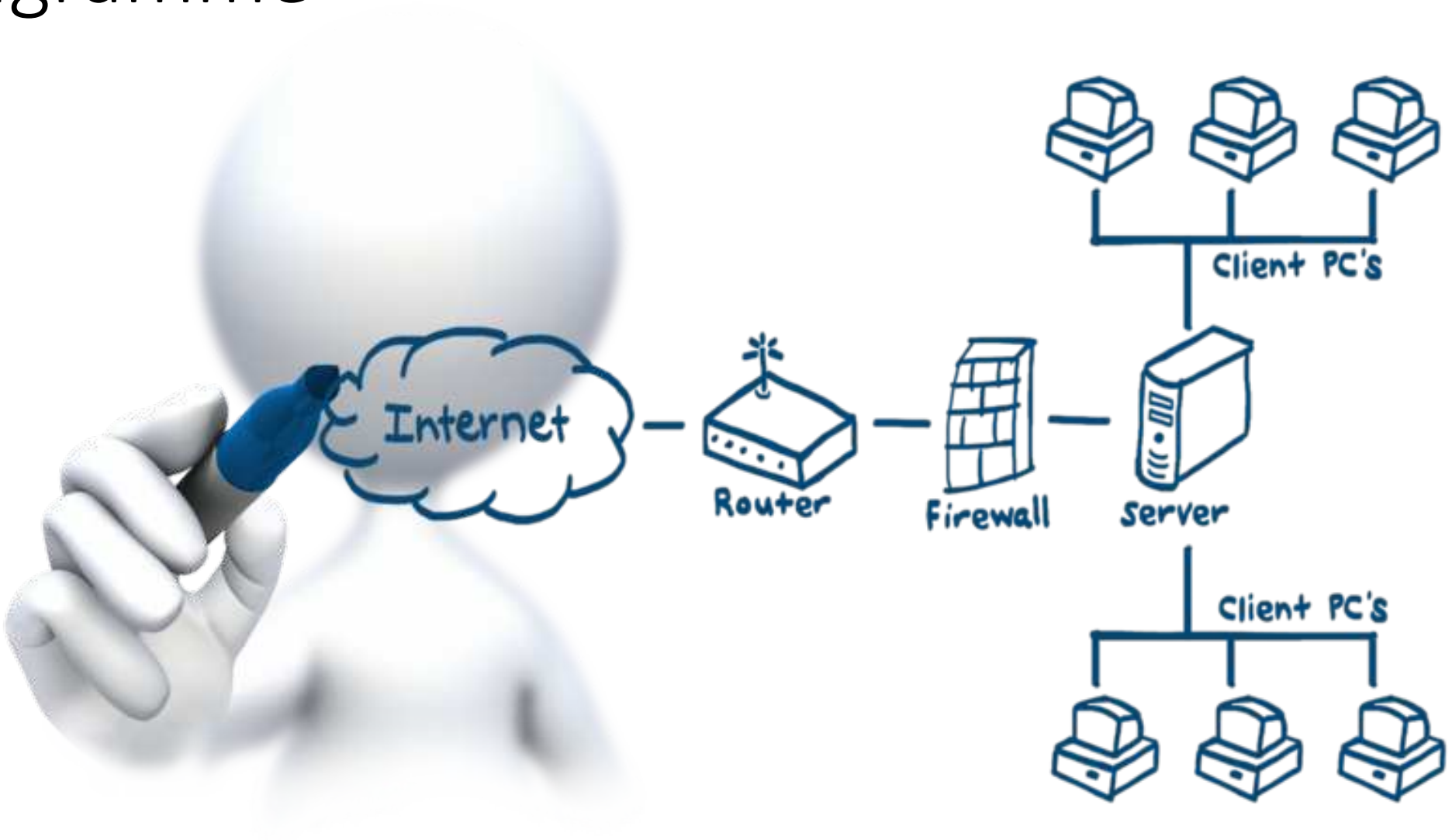
Tabellen!?


Requirements.xlsx
Ralf Müller
RM
File
Home
Insert
Draw
Page Layou
Formulas
Data
Review
View
Help
B6
A
B
1
Akteur
Beschreibung
2
App-Nutzer/in
Erhält Informationen über mögliche Begegnungen mit infizierten Personen und eigene Testergebnisse. Verifiziert eigene Testergebnisse und warnt so freiwillig andere.
3
Verifikations-Hotline
Unterstützt App-Nutzer/innen bei der Freischaltung positiver Testergebnisse (“teleTAN").
4
Gesundheitsämter und Testlabore
Liefern anonymisierte Testergebnisse an das System.
5
Ausländische Kontaktverfolgungen
Austausch mit dezentralen Anwendungen anderer Länder zur grenzüberschreitenden Ermittlung von Kontakten.
6
7
Systemkontext
Qualitätszi ...
Display Settings
100 %

== exportExcel

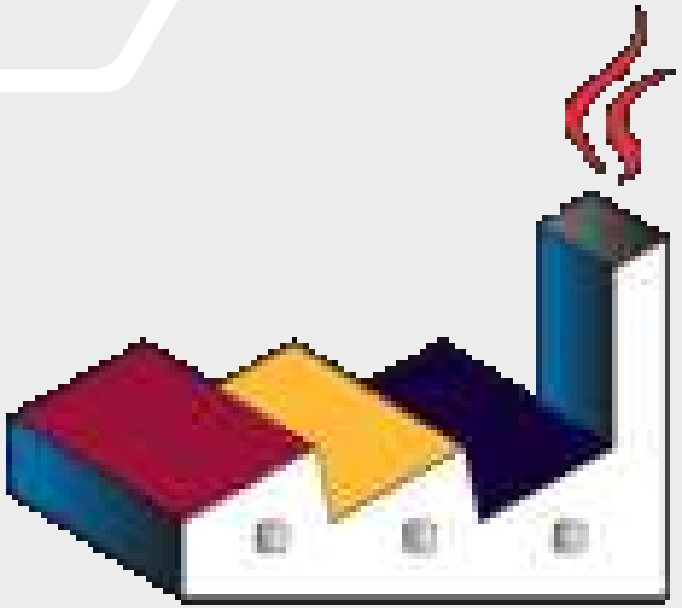
./dtcw exportExcel

Diagramme



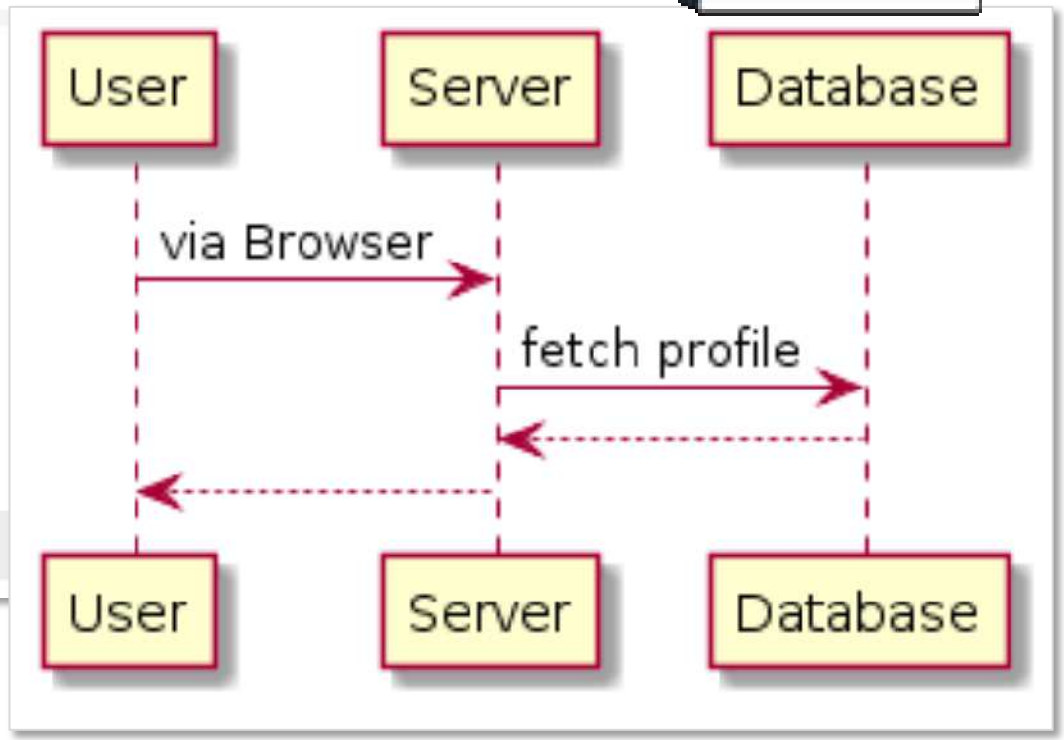
Diagramme

- http://plantuml.com/
- http://asciidoctor.org/docs/asciidoctor-diagram/
- Komplexe Diagramme als einfachen Text verwalten



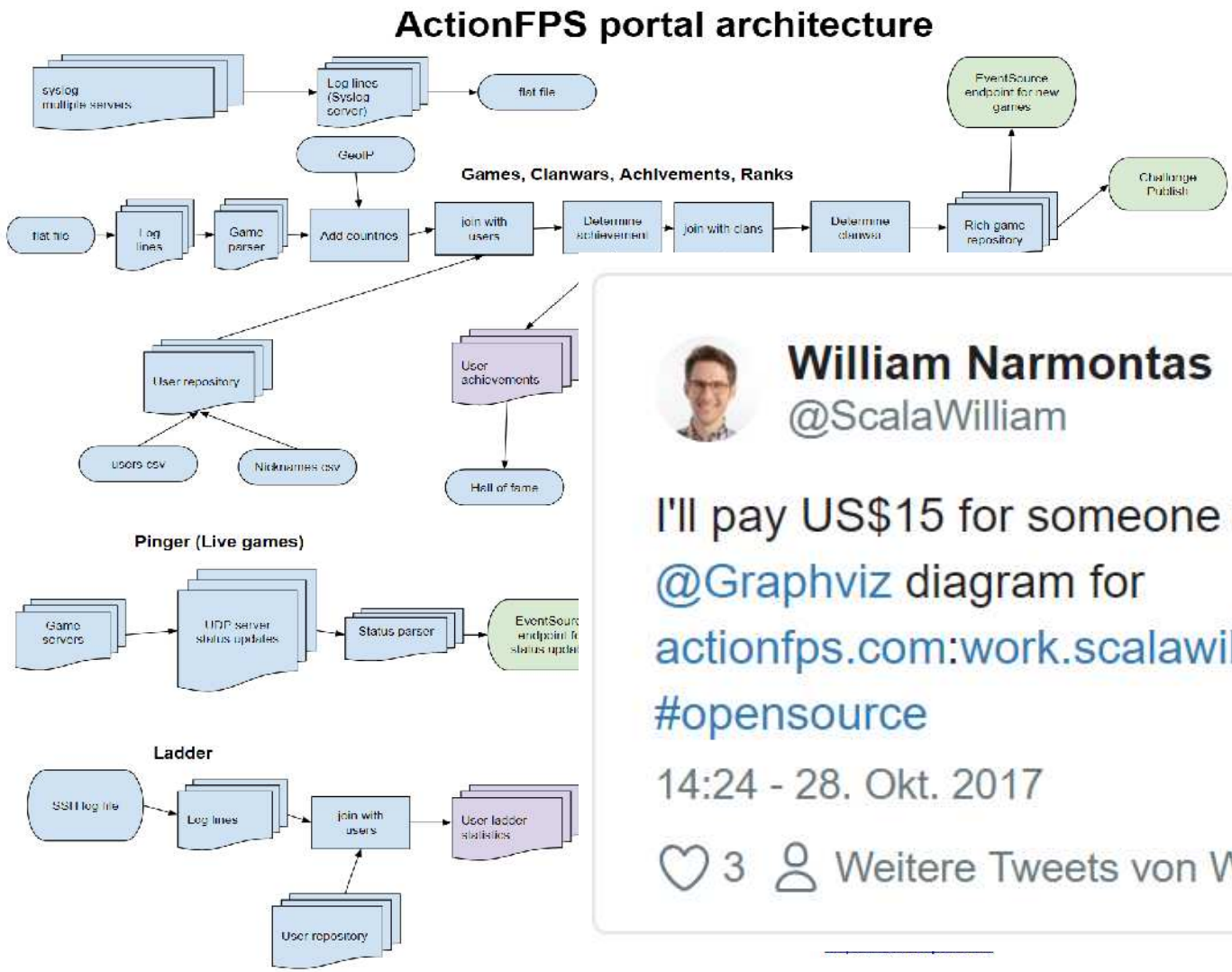
PlantUML


```
1  @startuml
2
3  User ->  Server: via Browser
4    Server ->  Database: fetch profile
5    Server <-- Database
6  User <-- Server
7
8  @enduml
```



./src/docs/diagrams.adoc

Diagramme: PlantUML



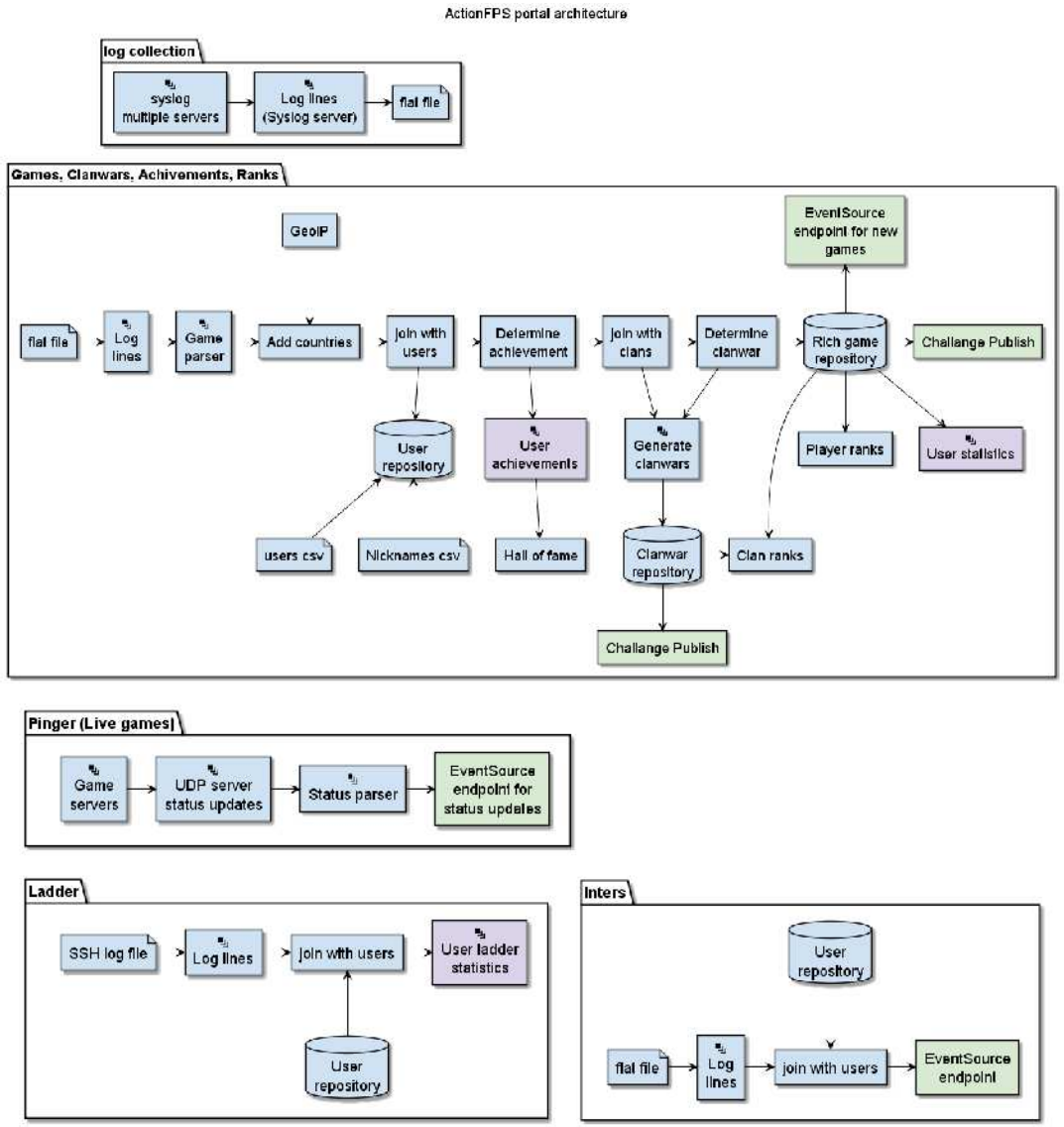
William Narmontas
@ScalaWilliam

I'll pay US$15 for someone who can do this @PlantUML or @Graphviz diagram for actionfps.com:work.scalawilliam.com/graphviz-plant… #opensource

14:24 - 28. Okt. 2017

3 Weitere Tweets von William Narmontas ansehen

Diagramme: PlantUML



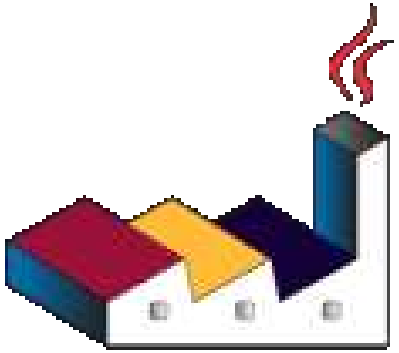
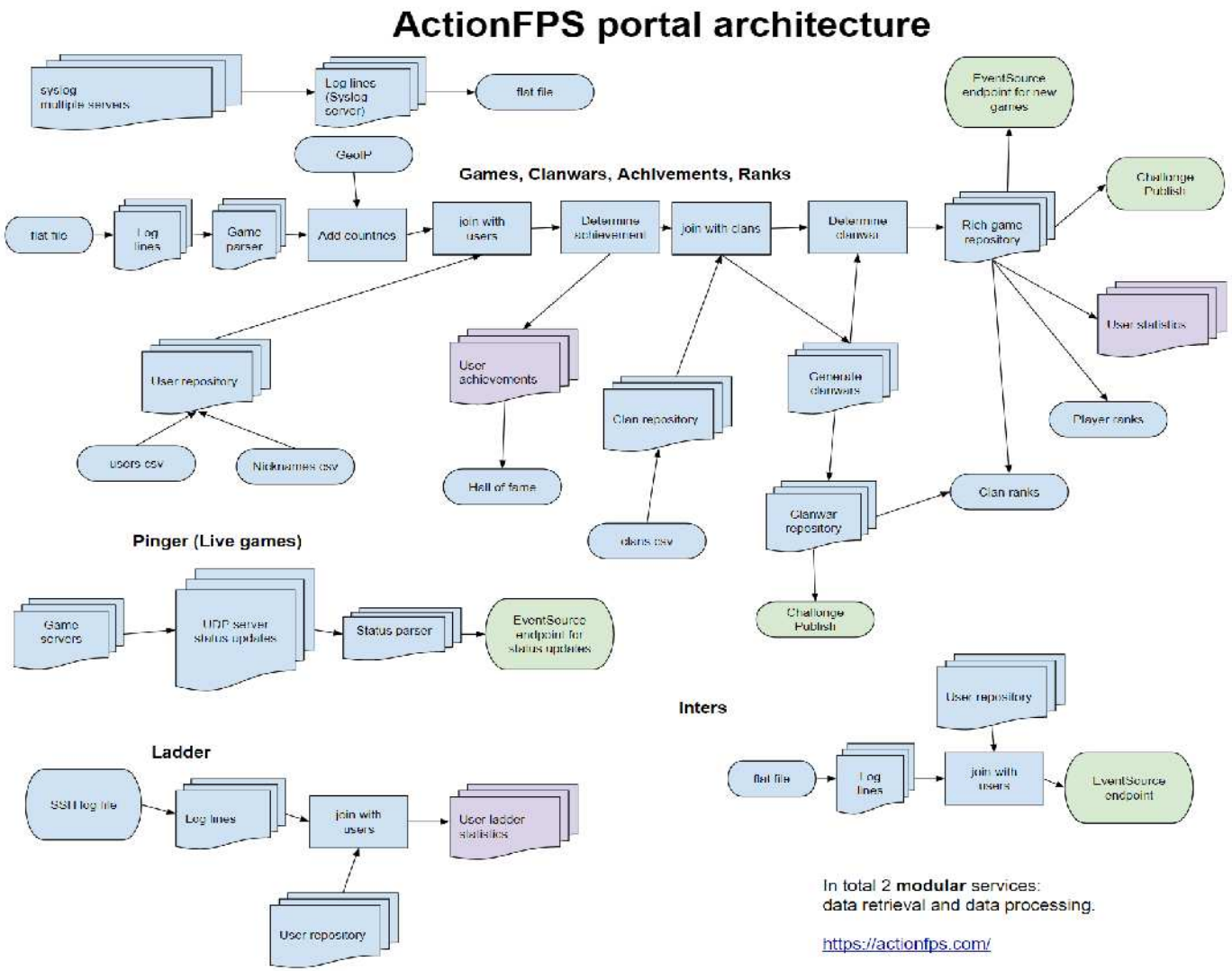


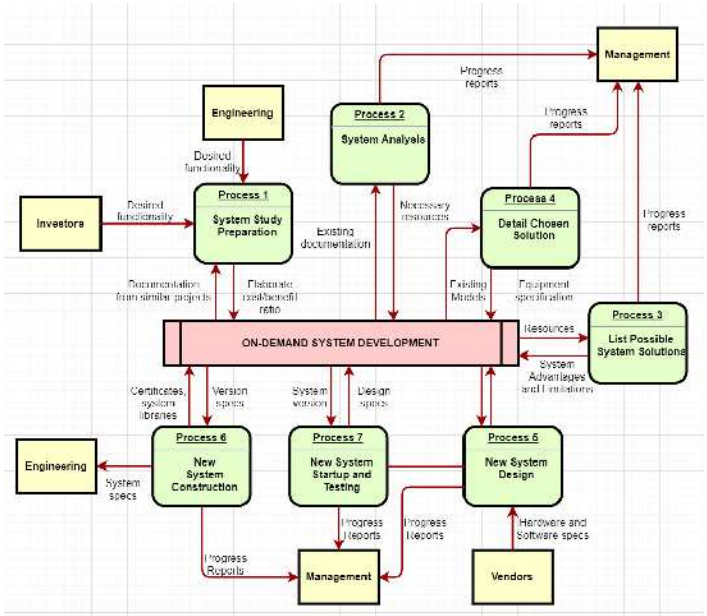
Diagramme: PlantUML



In total 2 **modular** services:
data retrieval and data processing.

https://actionfps.com/

Diagrams.net



<xml>

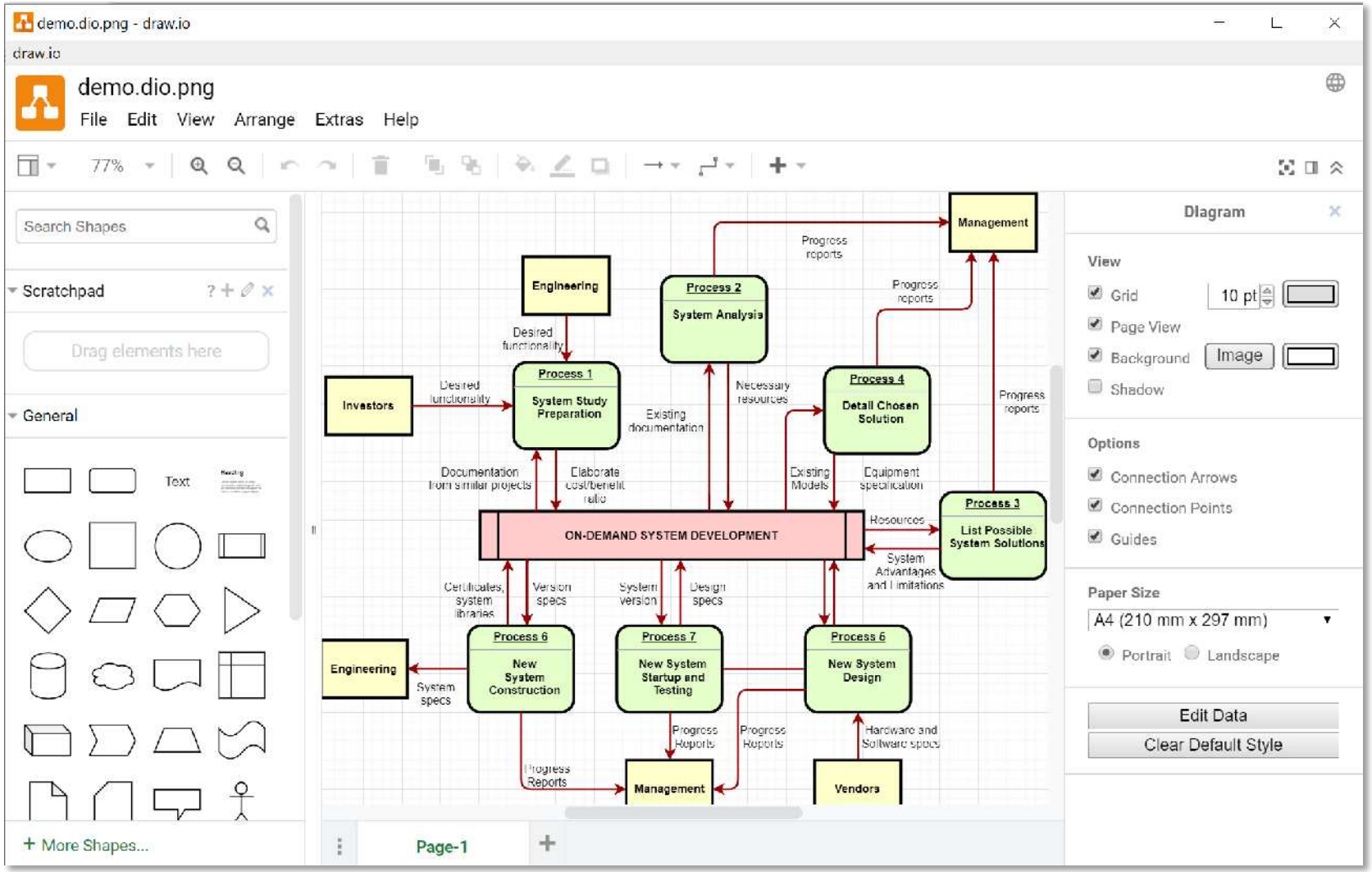
…

</xml>

PNG-Diagramm

Meta-Daten

Diagrams.net



./src/docs/images/demo.dio.svg

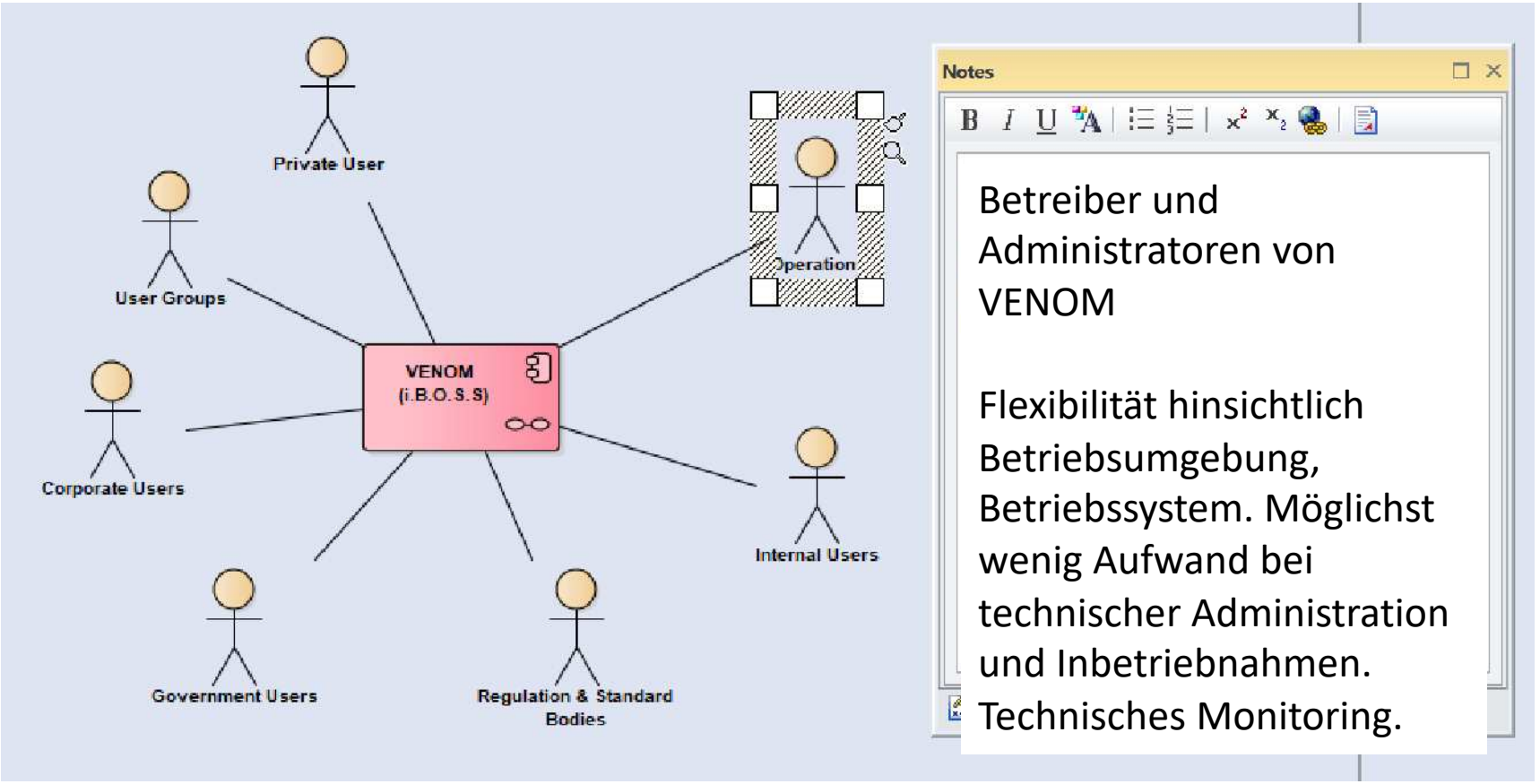
Kurze Pause bis 11:30 Uhr

Agenda

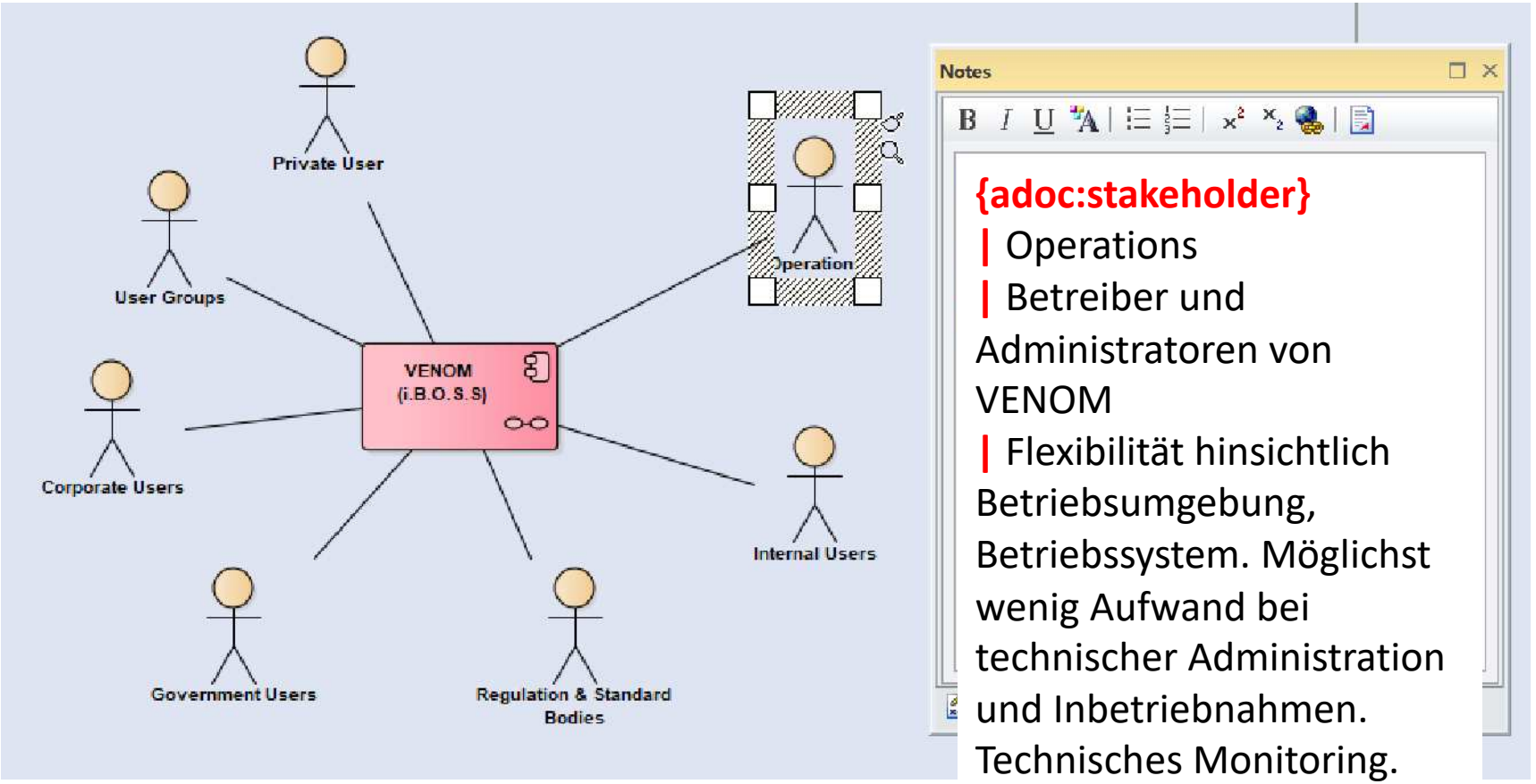
- Einführung
- Documentation as Code
- docToolchain
- Architektur dokumentieren
- Praktische Umsetzung
- **Weiterführende Themen**
- Fazit und Ausblick

Diagramme: Nicht malen, modellieren!

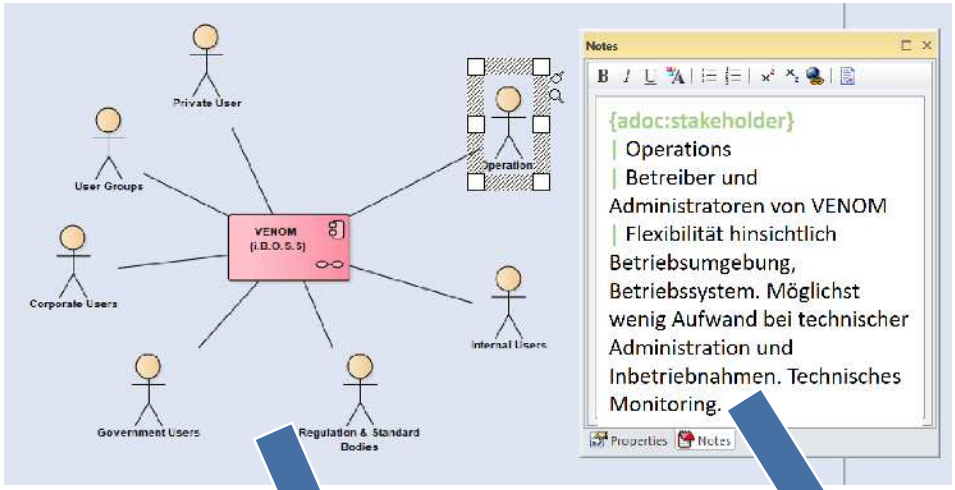
treat Docs-as-Code: automate!

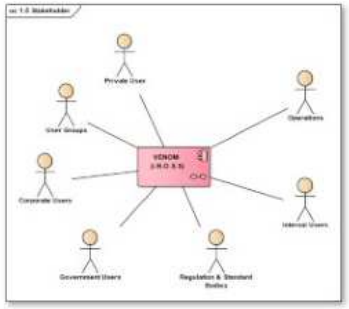


treat Docs-as-Code: automate!



treat Docs-as-Code: automate!



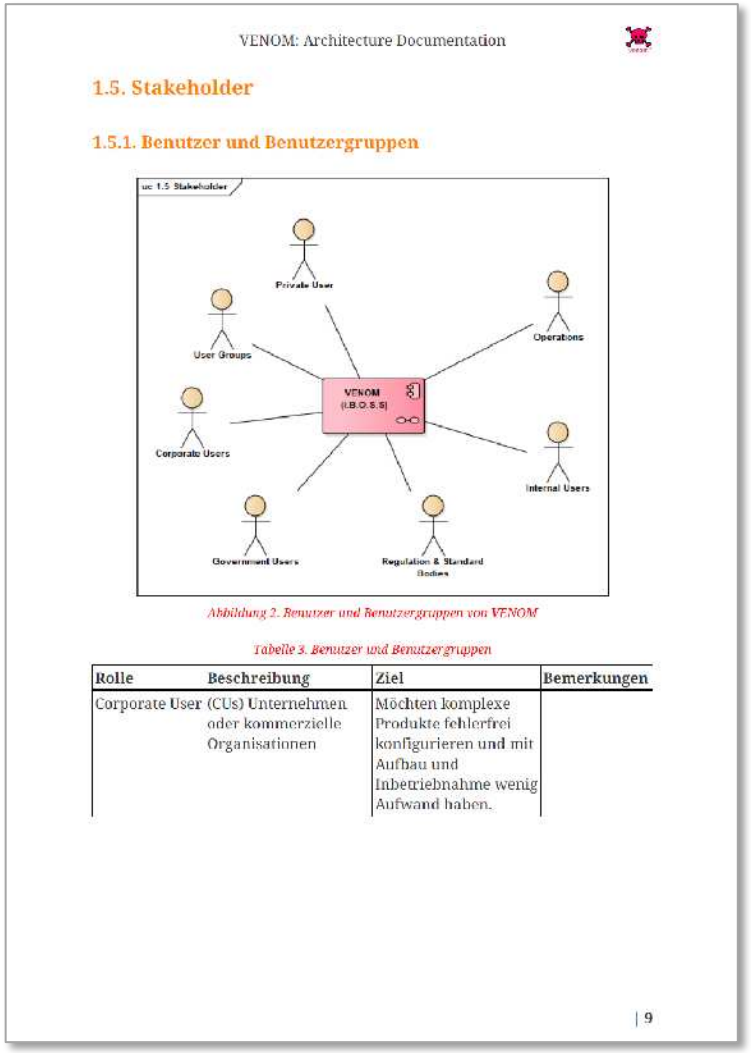


1.5_Stakeholder.png

stakeholder.ad

```
=== Stakeholder

==== Users and Groups of Users


image::ea/1.5_Stakeholder.png[]

[cols="2,3,3,2" options="header"]
.Users and Groups of Users
|===
| Role | Description | Goal  | Comment

include::../../ea/stakeholder.ad[]

|===
```

treat Docs-as-Code: automate!



VENOM: Architecture Documentation

1.5. Stakeholder

1.5.1. Benutzer und Benutzergruppen

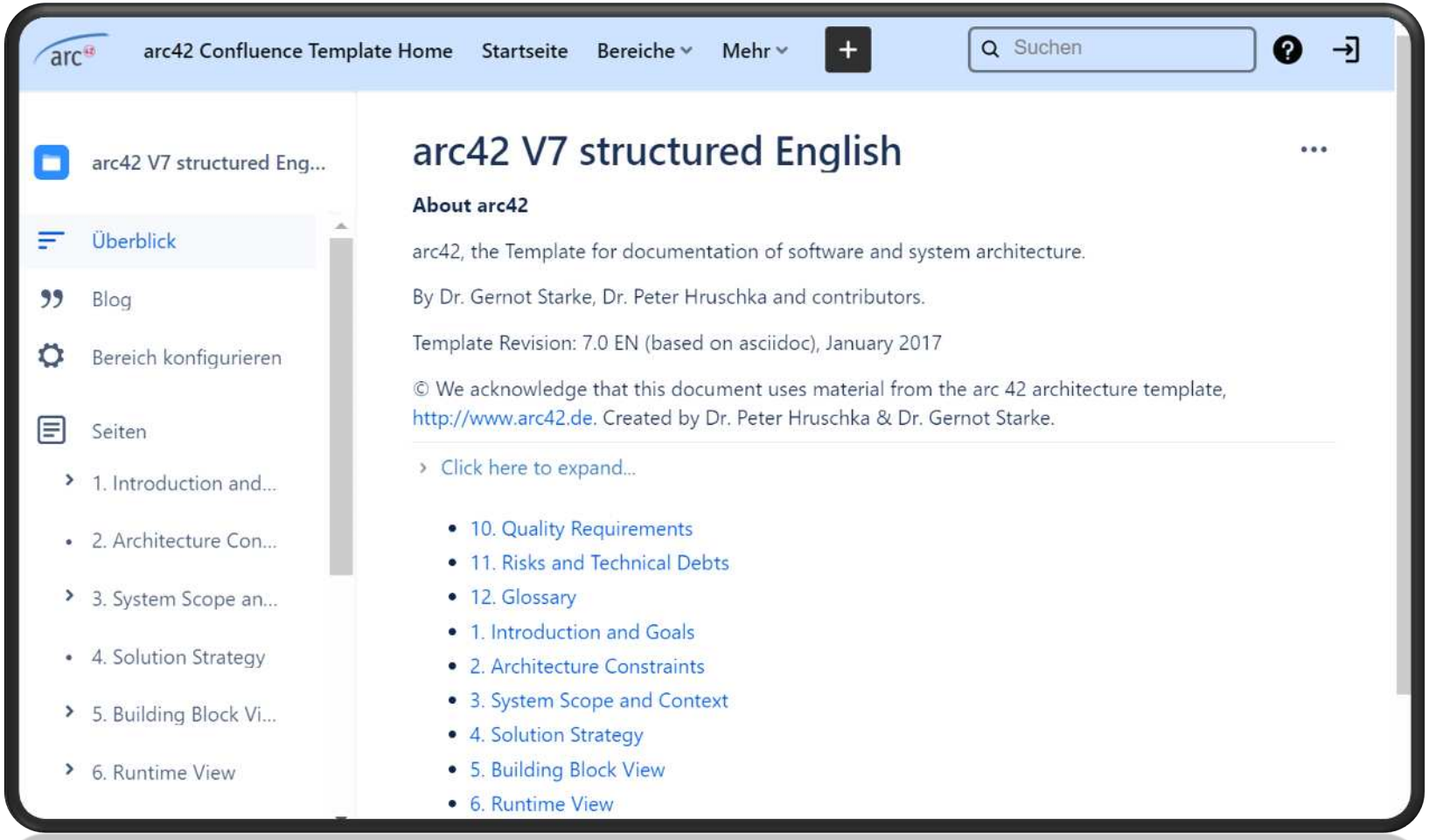
Abbildung 2. Benutzer und Benutzergruppen von VENOM

Tabelle 3. Benutzer und Benutzergruppen

Rolle	Beschreibung	Ziel	Bemerkungen
Corporate User	(CUs) Unternehmen oder kommerzielle Organisationen	Möchten komplexe Produkte fehlerfrei konfigurieren und mit Aufbau und Inbetriebnahme wenig Aufwand haben.	

| 9

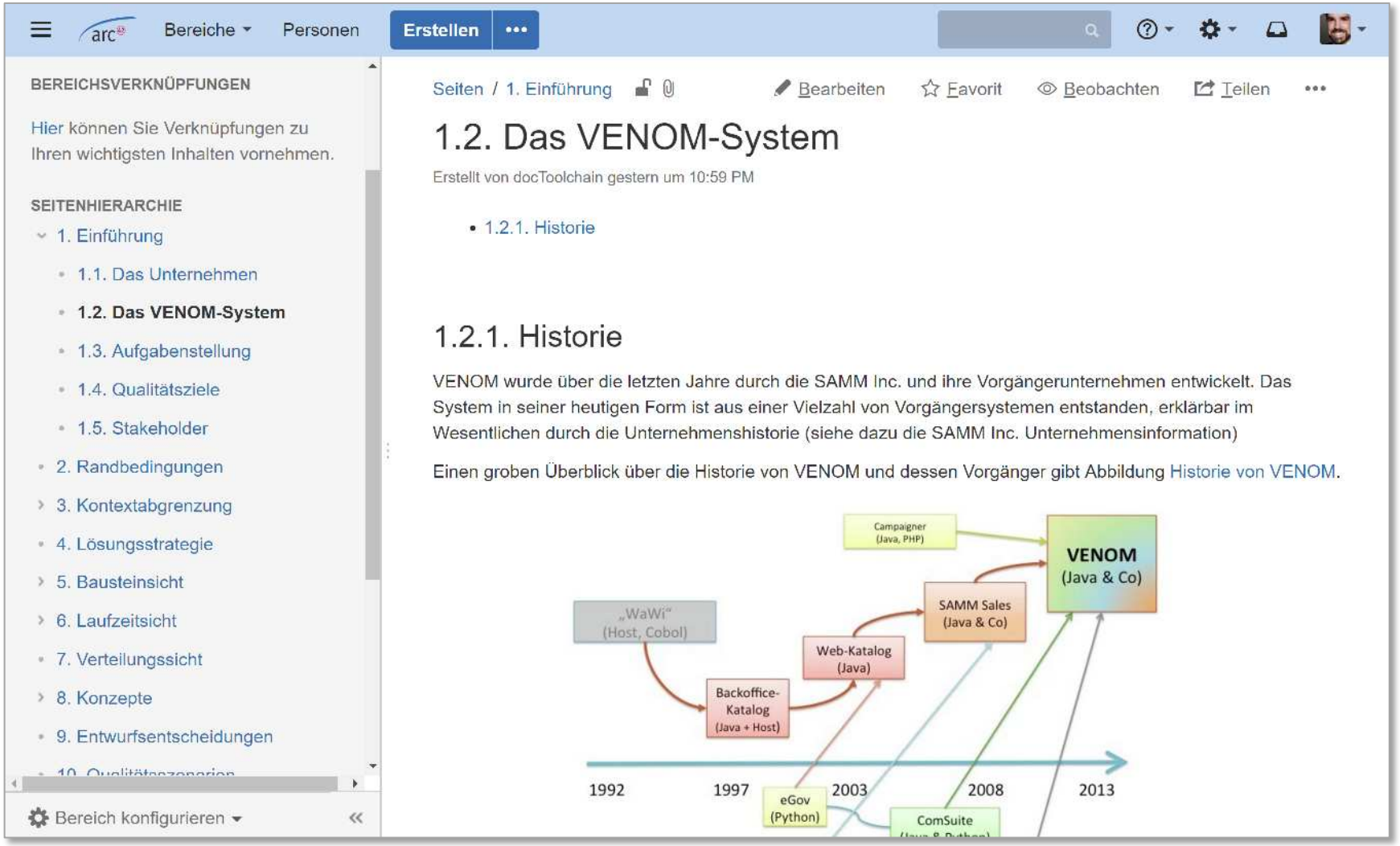
publishToConfluence



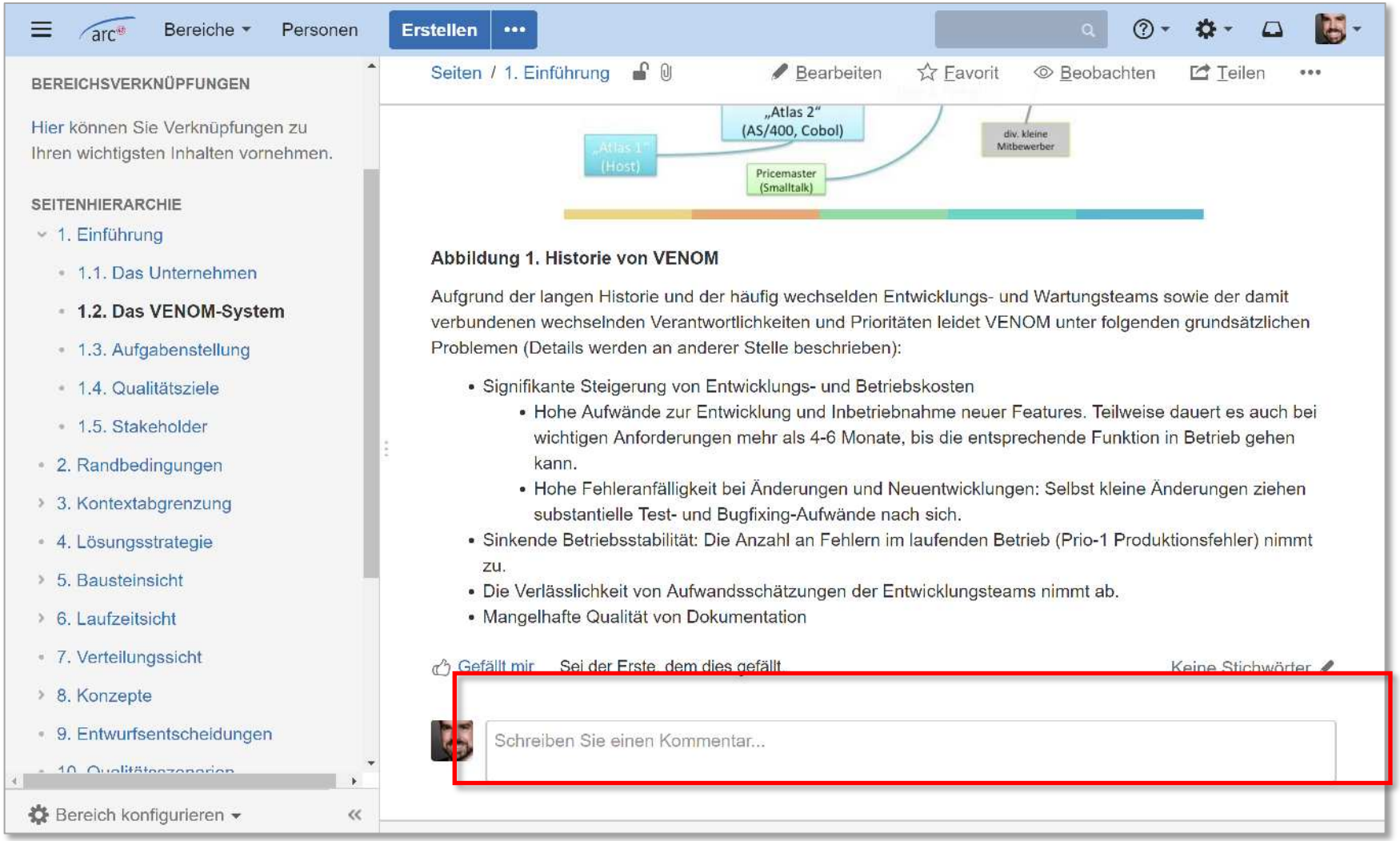
https://arc42-template.atlassian.net/wiki/spaces/7SE/overview

Zusammenarbeit

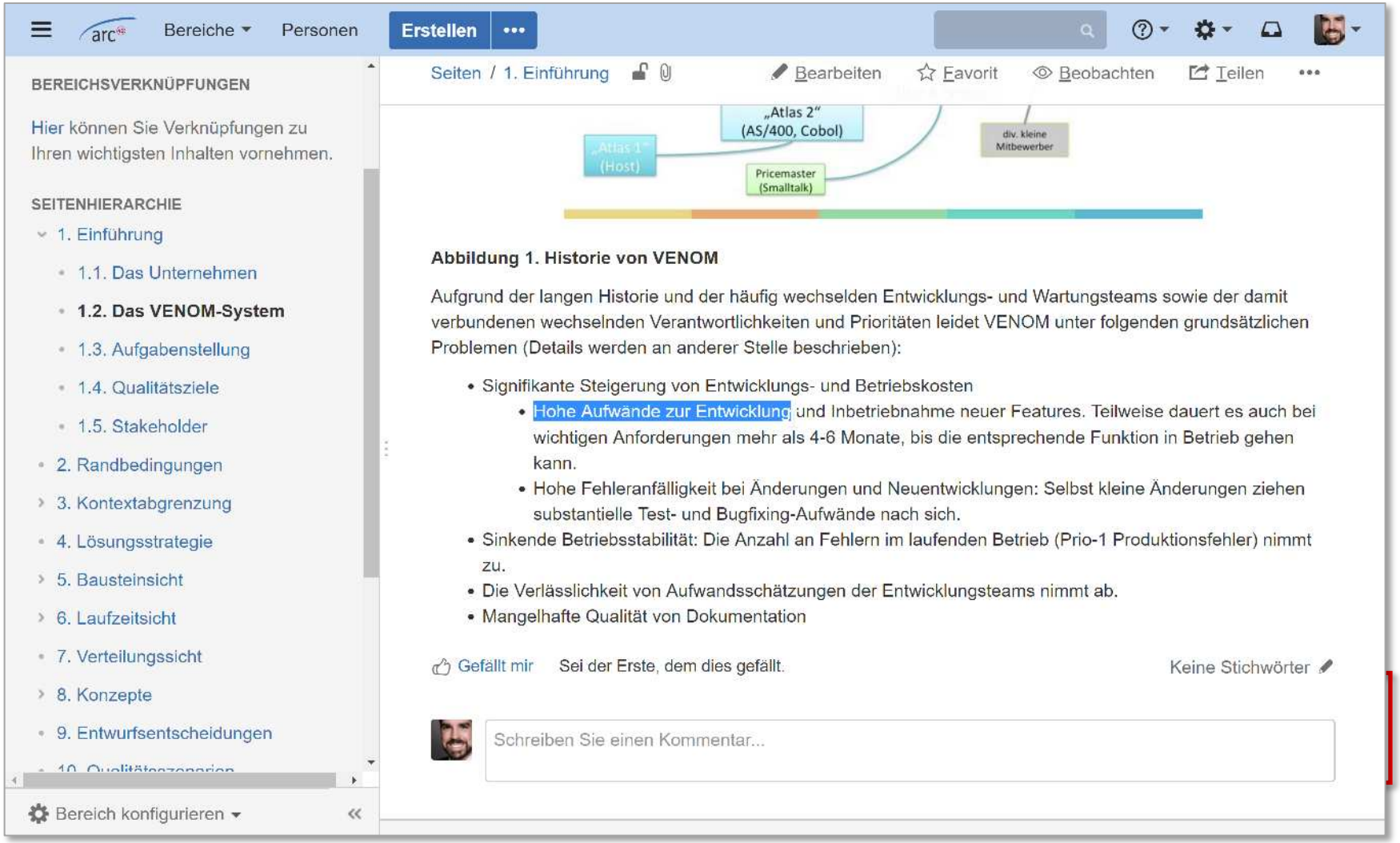
Zusammenarbeit



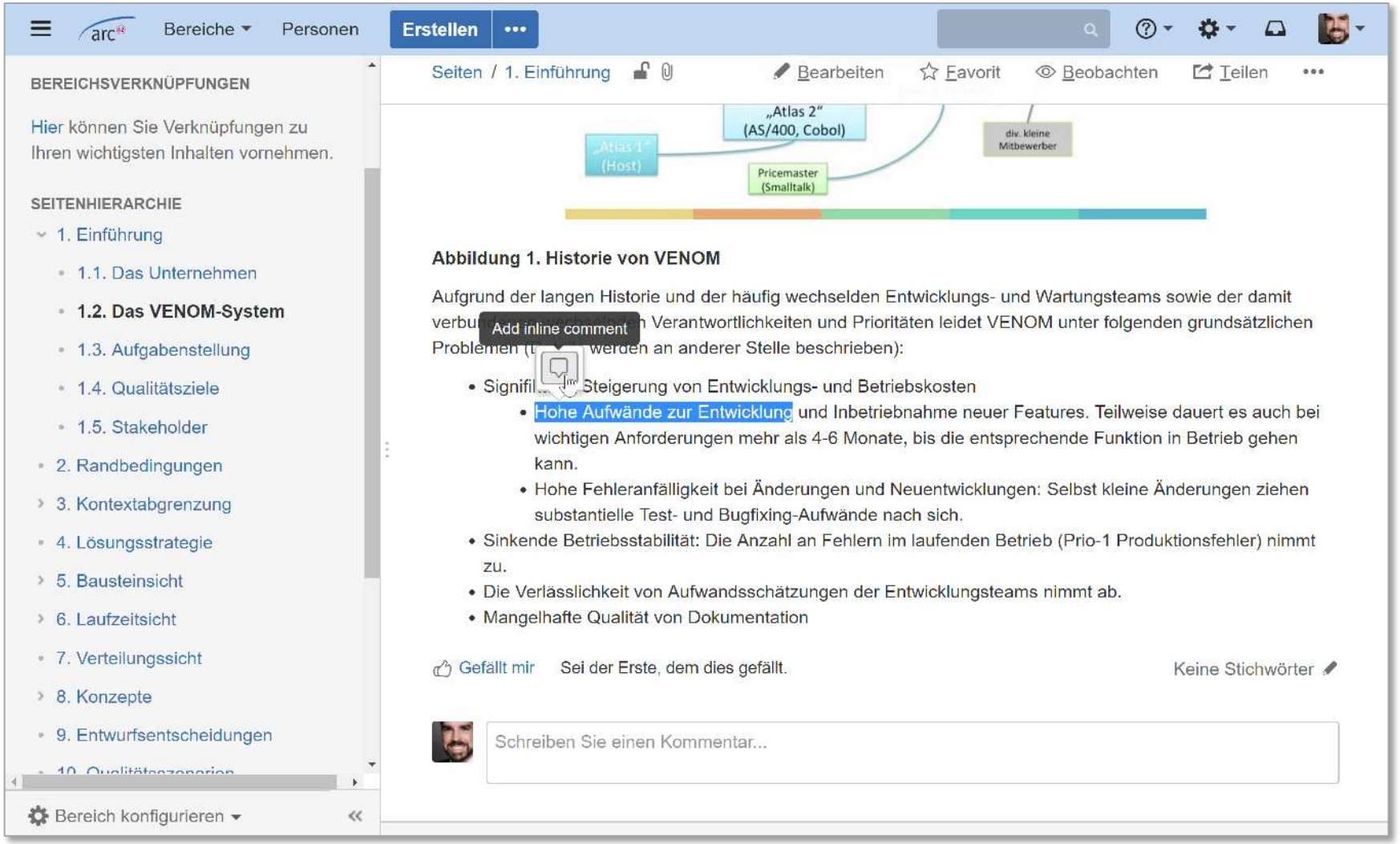
Zusammenarbeit



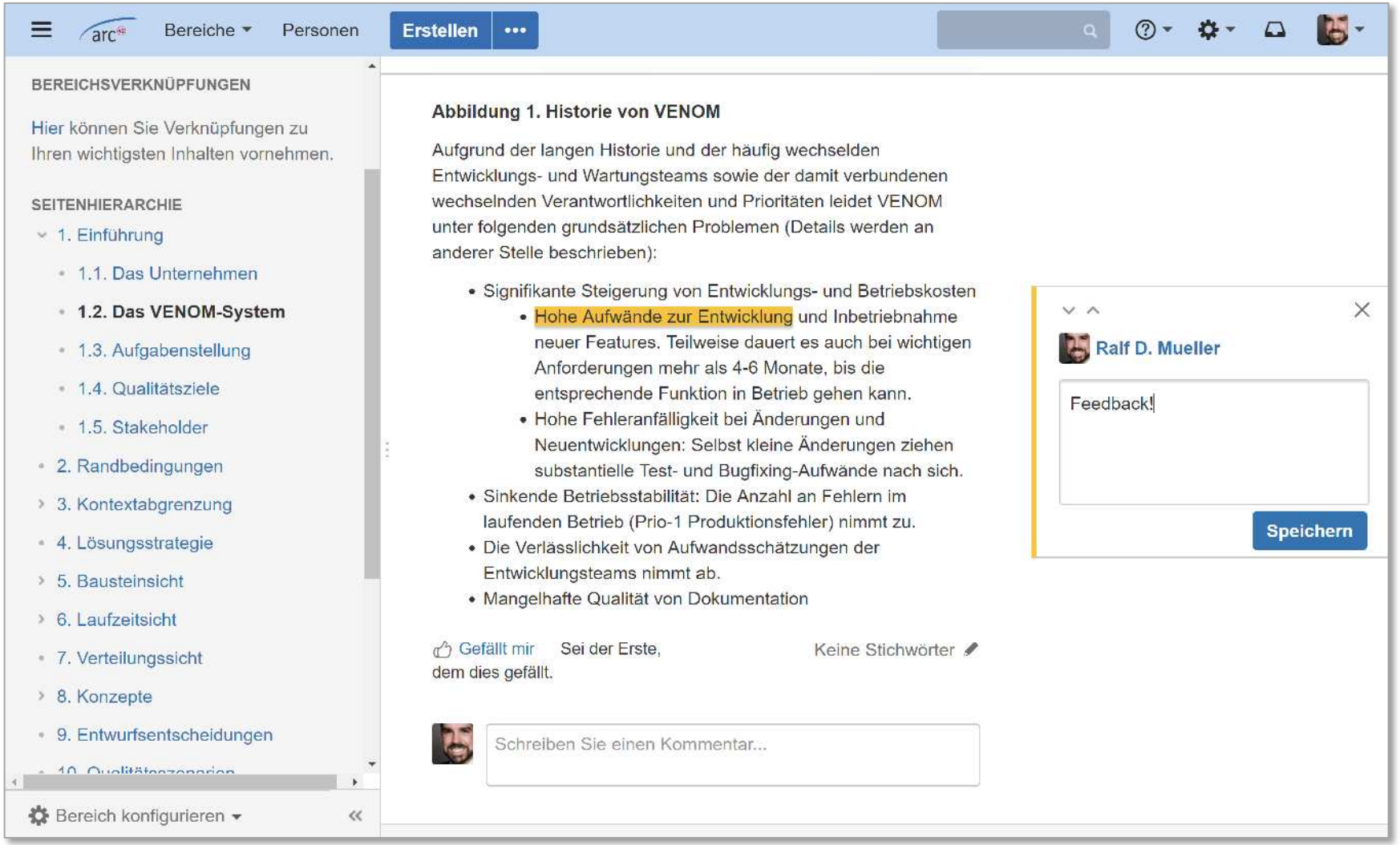
Zusammenarbeit



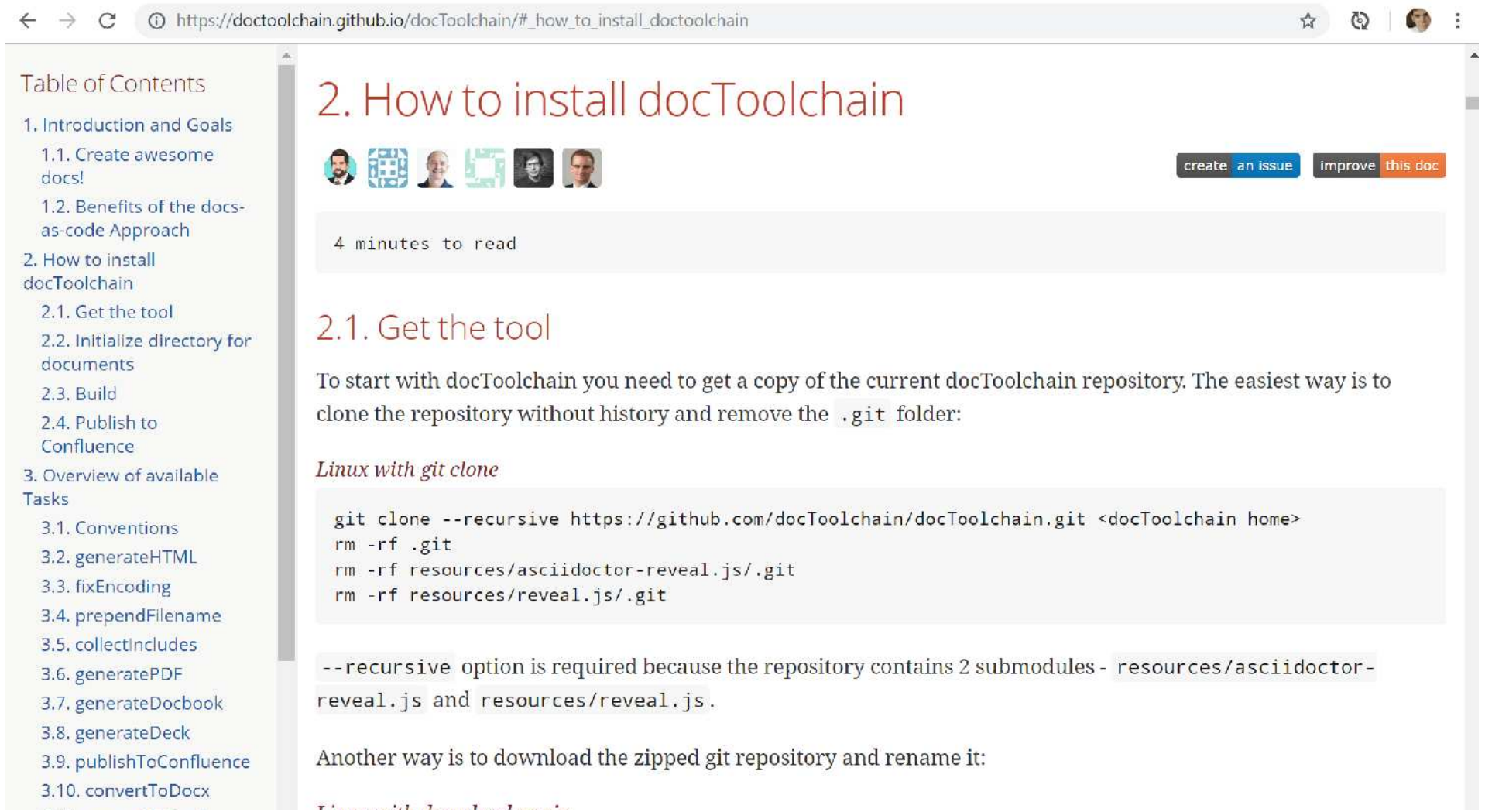
Zusammenarbeit



Zusammenarbeit



Besser: statische Seiten

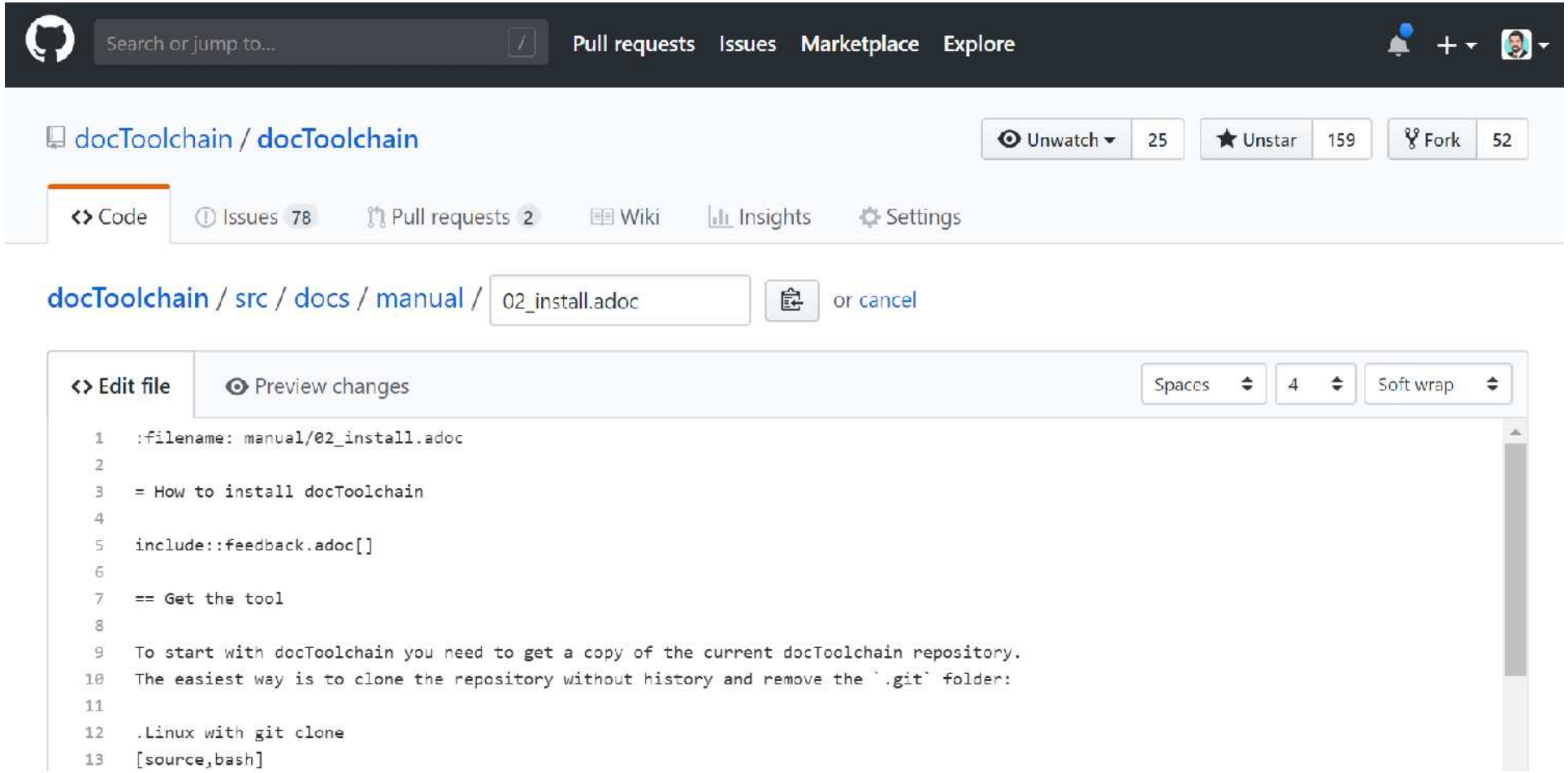


Besser: statische Seiten

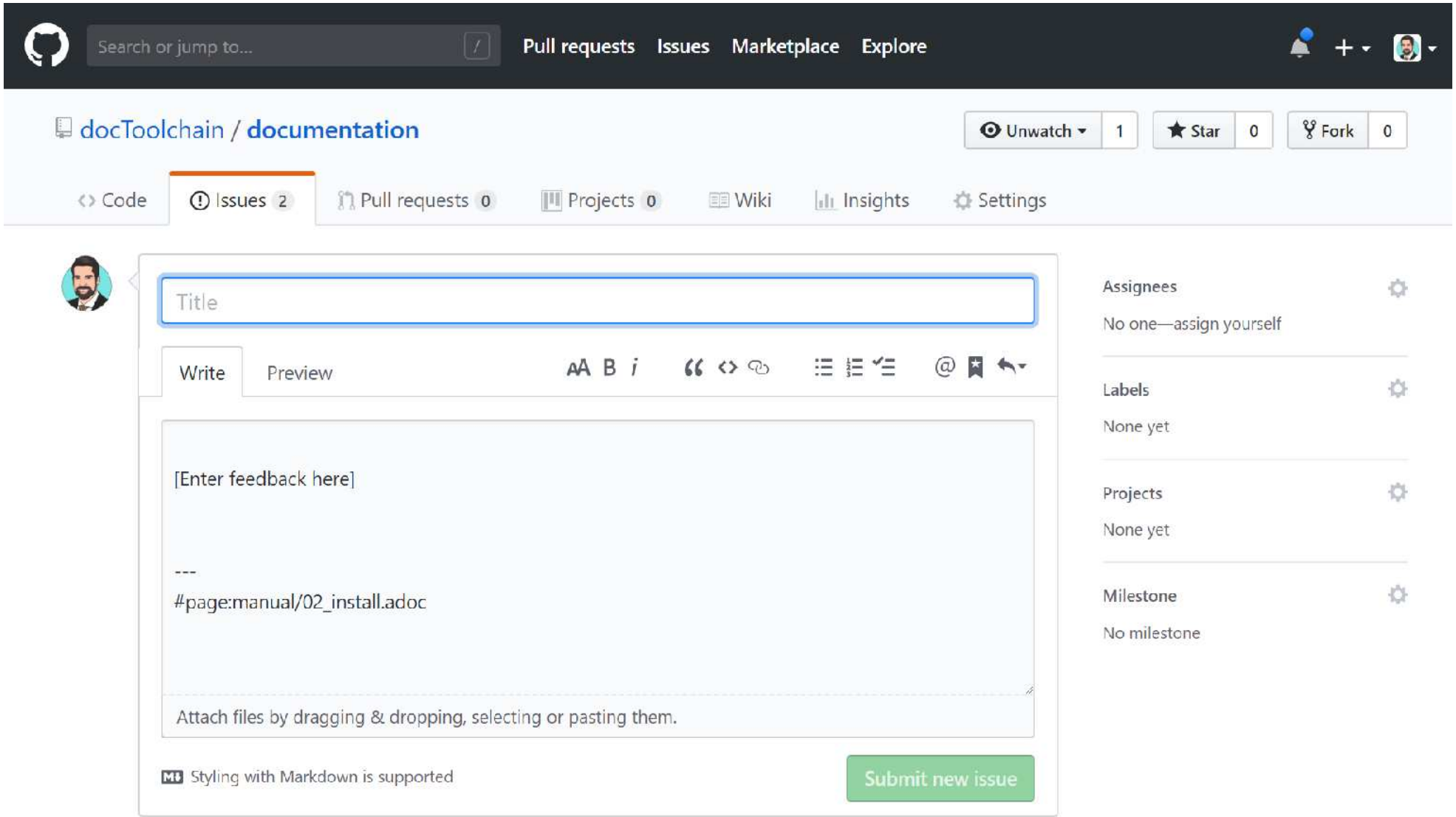
of the current docToolchain repository. The easiest way is to

he `.git` folder:

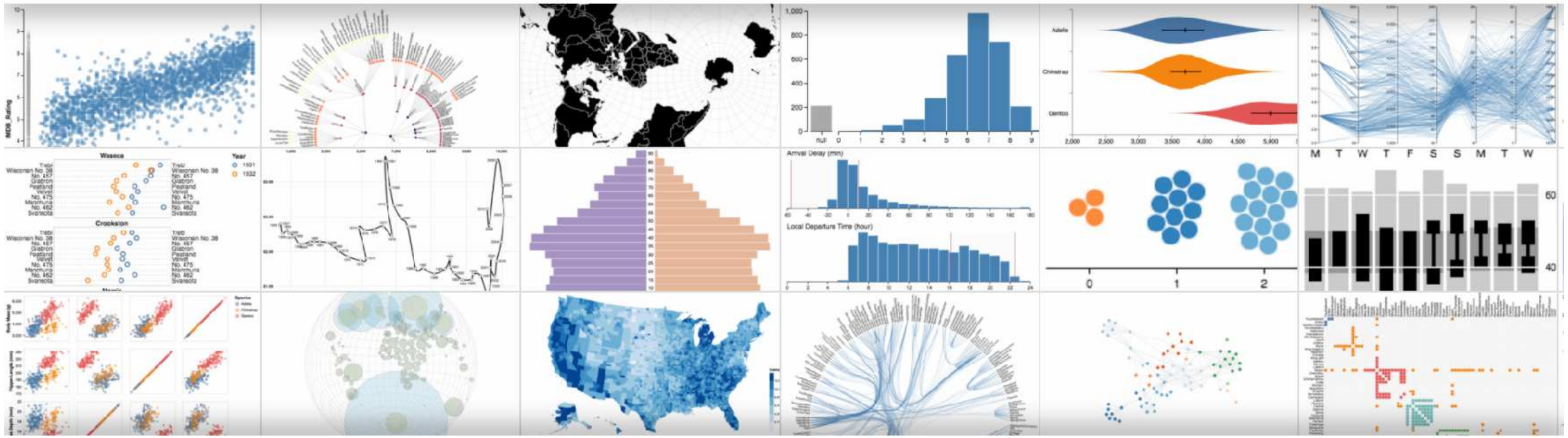
Besser: statische Seiten - improve this doc



Besser: statische Seiten - create an issue



Vega – A Visualization Grammar



https://vega.github.io/vega/

Kroki

Creates **diagrams** from **textual** descriptions!

Kroki provides a unified API with support for BlockDiag (BlockDiag, SeqDiag, ActDiag, NwDiag, PacketDiag, RackDiag), BPMN, Bytefield, C4 (with PlantUML), Ditaa, Erd, Excalidraw, GraphViz, Mermaid, Nomnoml, PlantUML, SvgBob, UMLet, Vega, Vega-Lite, WaveDrom... and more to come!

#Features

Ready to use

Diagrams libraries are written in a variety of languages: Haskell, Python, JavaScript, Go, PHP, Java... some also have C bindings. Trust us, you have better things to do than install all the requirements to use them. Get started in no time!

Simple

Kroki provides a unified API for all the diagram libraries. Learn once use diagrams anywhere!

Free & Open source

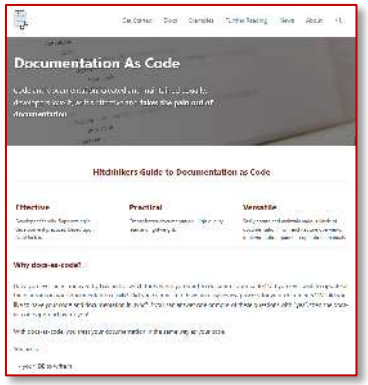
All the code is available on GitHub and our goal is to provide Kroki as a free service.

Fast

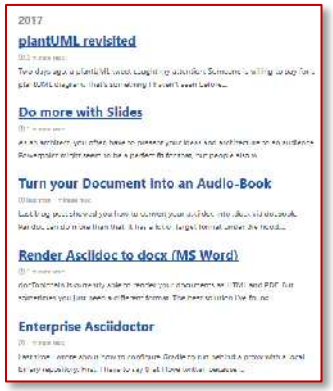
Built using a modern architecture, Kroki offers great performance.

https://kroki.io/

Static Site Generators



Landing-Page



Blog

arc42 Dokumentation

User-Manual

Test-Reports

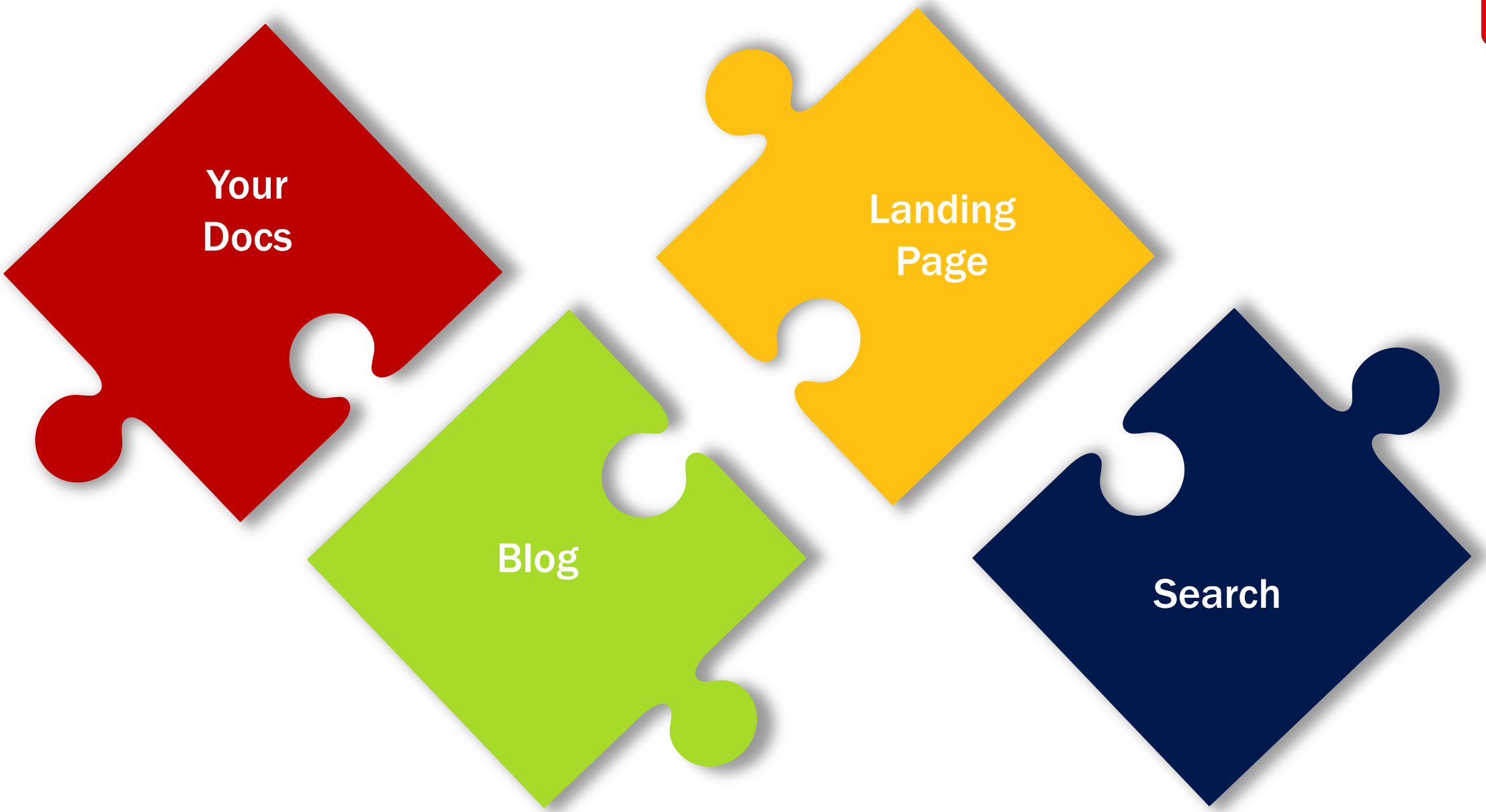
Suche

Als **Microsite** (bzw. Mikro-Website) bezeichnet man im Webdesign eine **schlanke Website** mit wenigen Unterseiten und geringer Navigationstiefe innerhalb eines größeren Internet-Auftritts. Die Microsites sind optisch von der eigentlichen Website unabhängig und bilden **thematisch und gestalterisch eine eigenständige kleine Internetpräsenz**.

Intranet

https://de.wikipedia.org/wiki/Microsite

Elemente einer Microsite



Elemente einer Microsite

== generateSite

./dtcw generateSite

Landing-Page

DB Microsite-Master Tutorial Architektur News About Search DB Systel Digital bewegen. Gemeinsam.

Microsite Master

Diese Microsite hilft Dir und Deinem Team eine eigene Microsite für Dokumentation mit Hilfe des Docs-as-Code Ansatzes aufzubauen.

Das Aushängeschild für Dein Projekt

Erstelle eine Microsite für die Kunden Deines Projekts. Von Engineers - für Engineers. Benutze dabei die über den Docs-as-Code Ansatz die gleichen Tools, die Du auch für's Coding verwendets.

arc42 Template

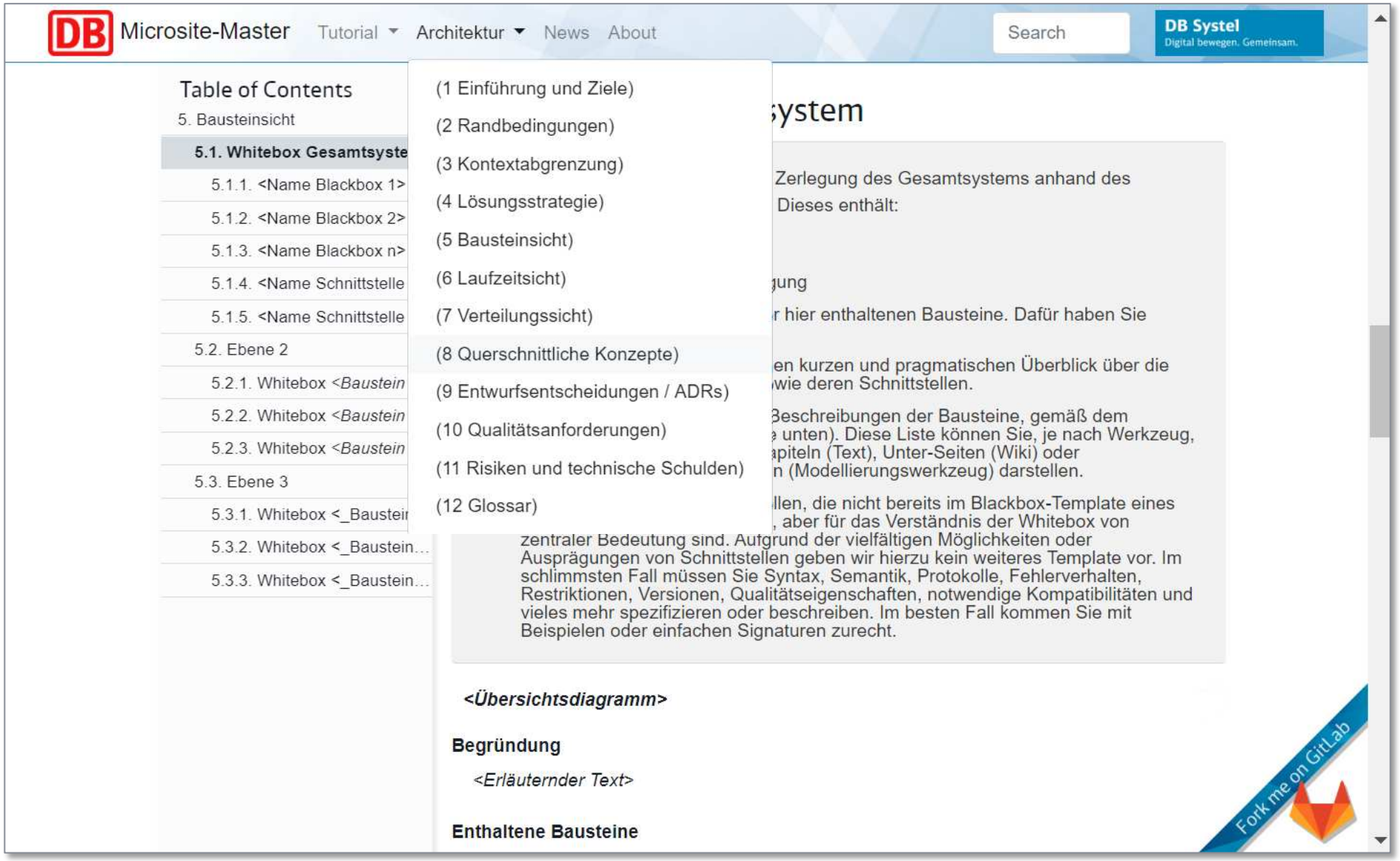
Architekturdokumentation einfach. Das vorkonfigurierte arc42-Tempalte dient als "Kleiderschrank" für Deine Dokumentation. Jeder Aspekt Deiner Architektur hat hier seinen Platz.

DB-Search Integration

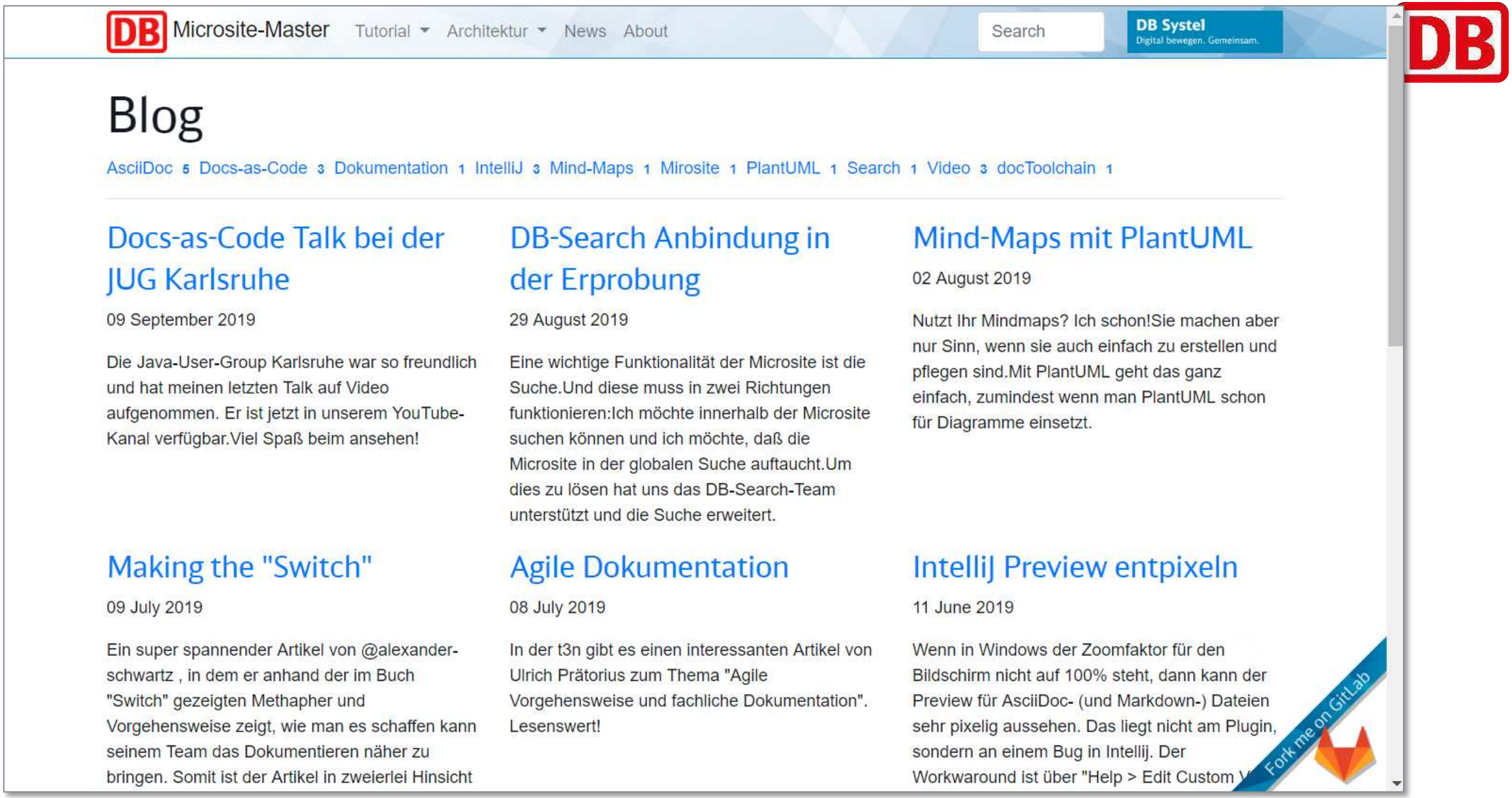
Nicht nur finden, sondern auch gefunden werden. Durch die Integration von DB-Search lässt sich die Microsite durchsuchen, wird abe auch bei anderen Suchanfragen auf DB Se gefunden!

Fork me on GitLab

Umfangreiche AsciiDoc-Dokumentation



News / Blog mit RSS-Feed



News / Blog mit RSS-Feed

Asciidoctor Plugin für IntelliJ

05 June 2019 | Tags: AsciiDoc IntelliJ

Ralf D. Müller ist Mitglied der Software Engineering Advocates und der Architektur Gilde. Seine Spezialgebiete sind Dokumentation (Docs-as-Code, arc42) und Security

Gestern wurde die neueste Version des Plugins von @alexander-schwartz veröffentlicht. Gerade die letzten Updates haben extrem viele neue Features reingebracht, die es Dir nun erlauben AsciiDoc Dokumente tatsächlich auch wie Code zu bearbeiten. Features wie Code-Folding, Strukturübersicht, Syntax-Highlighting und Autovervollständigung funktionieren nun auch in .adoc-Files. Praktisch sind auch die live-templates. Einfach mal ad tippen und schon erhältst Du eine Liste an Templates, die das Nachschlagen in der AsciiDoc-Dokumentation erübrigen.

Die vollständige Liste der Features findest Du auf GitHub:

https://github.com/asciidoctor/asciidoctor-intellij-plugin/blob/master/FEATURES.adoc

Agenda

- Einführung
- Documentation as Code
- docToolchain
- Architektur dokumentieren
- Praktische Umsetzung
- Weiterführende Themen
- **Fazit und Ausblick**

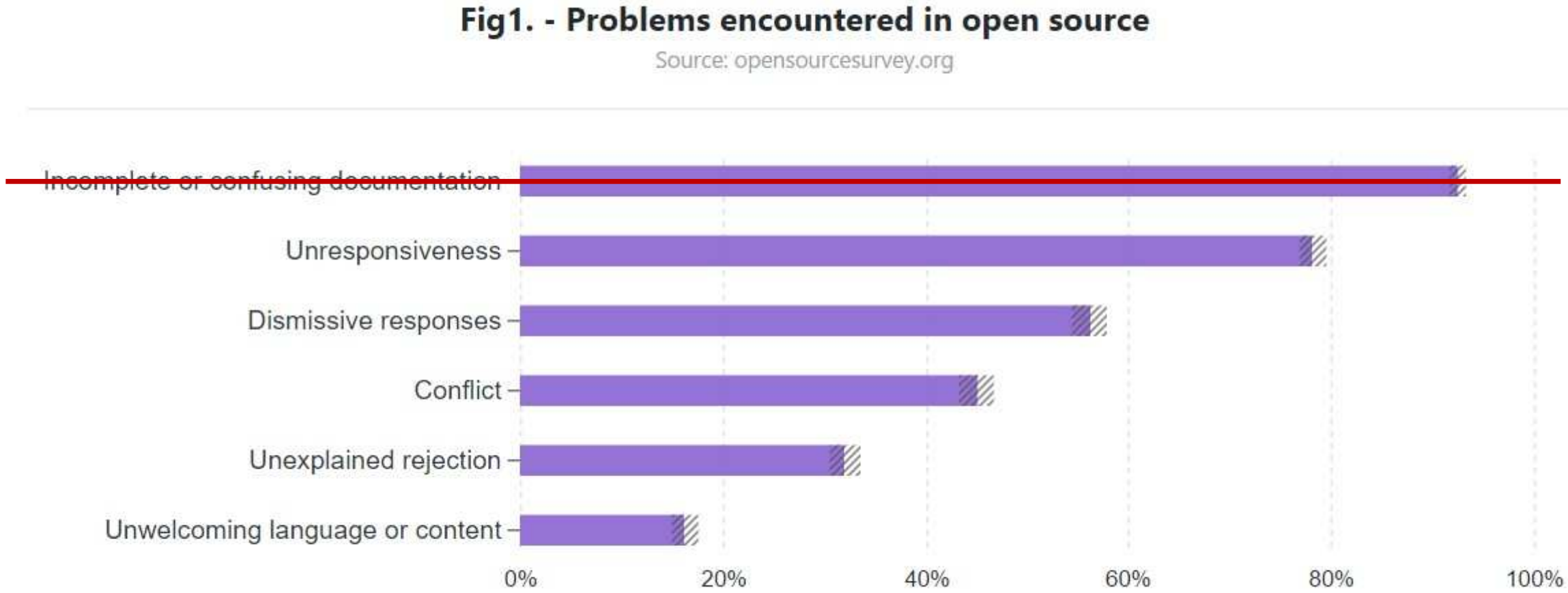


Fig1. - Problems encountered in open source
Source: opensourcesurvey.org
Incomplete or confusing documentation
Unresponsiveness
Dismissive responses
Conflict
Unexplained rejection
Unwelcoming language or content
0%
20%
40%
60%
80%
100%

Out-of-the-Box Features

„ablenkungsfrei“ – Dokumentation wie eMails schreiben

Gliederung in Unterdokumente

Neugliederung je nach Stakeholder

Bilder werden referenziert, nicht eingebettet

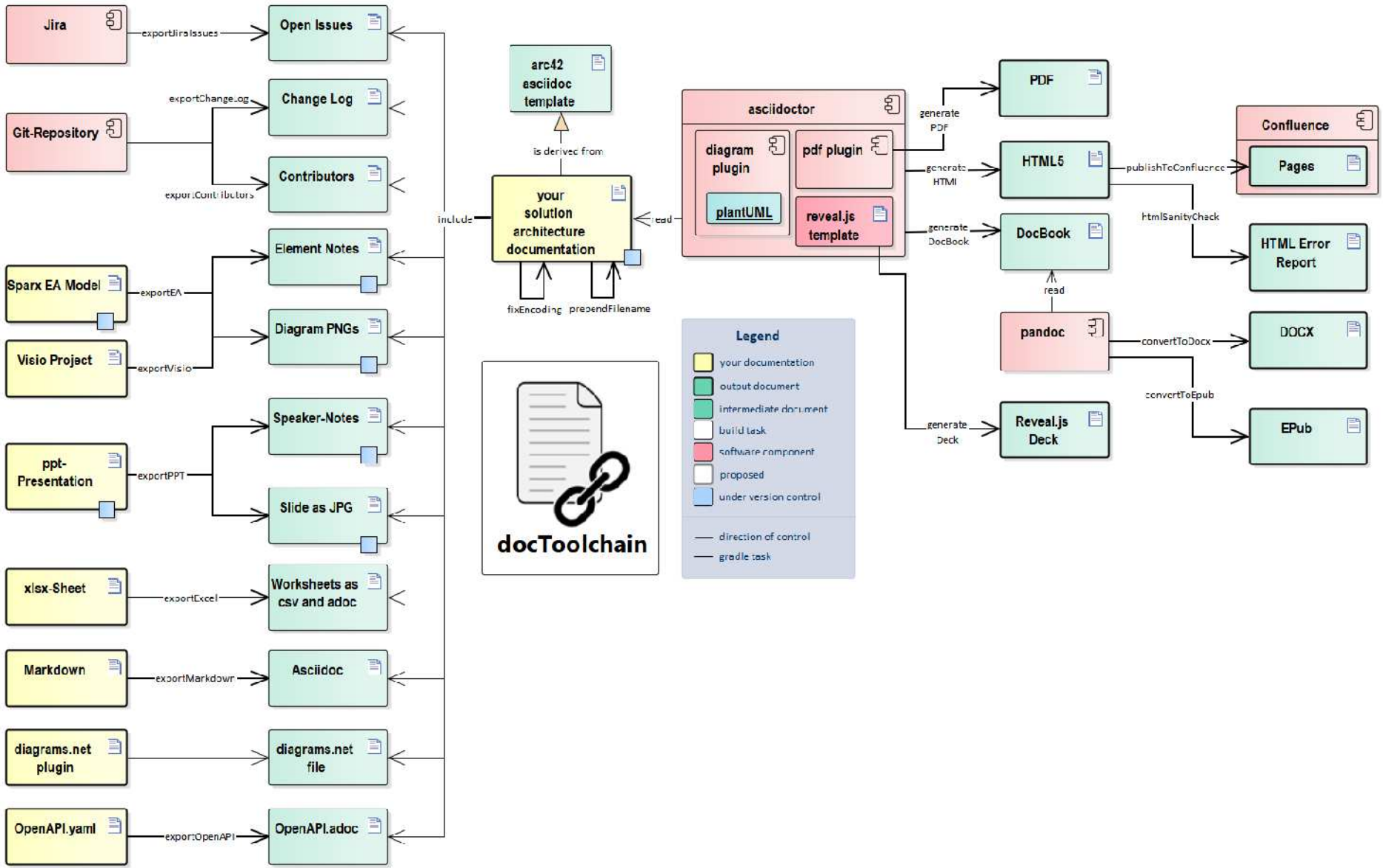
leichte Versionierung „handle Docs-as-Code“

Formatierung von Source-Code

Reviews, Pull-Requests, Versionierung durch Git

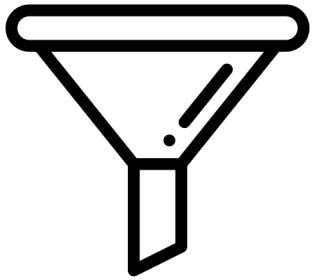
Konvertierung nach HTML5 und DocBook

docToolchain Übersicht

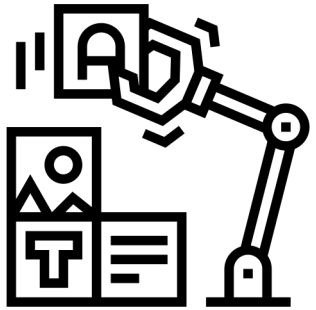


Moderne (Architektur-)Dokumentation

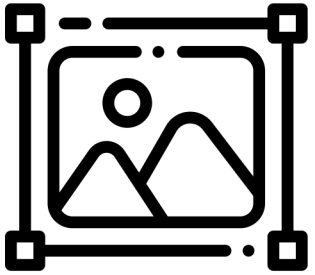
Textformate



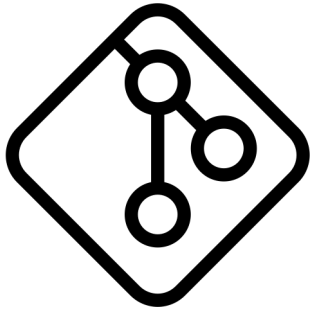
Single Source of Truth



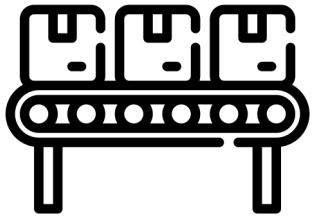
Inhalte generieren



Bilder/Grafiken



Docs-as-Code



Continuous Documentation

Ausführen/ Validieren

Kümmerer

Vielen Dank.

Wir freuen uns auf Eure Fragen!

falk.sippach@embarc.de

@sippsack

ralf.d.mueller@deutschebahn.com

@RalfDMueller