

Web Development in der Praxis

Theorie vs. Realität

```
350  
351  
352 /* =Menu  
353 -----  
354 .access {  
355   display: inline-block;  
356   height: 69px;  
357   float: right;  
358   margin: 11px 28px 0px 0px;  
359   max-width: 800px;  
360   position: relative;  
361   width: 100%;  
362 }  
363 .access .inner {  
364   background-color: #e6f2ff;  
365   border: 1px solid #d9e1f2;  
366   border-radius: 3px;  
367   color: #333399;  
368   font-size: 14px;  
369   font-weight: bold;  
370   padding: 10px 0px 0px 0px;  
371 }  
372 .access .inner a {  
373   color: inherit;  
374   text-decoration: none;  
375 }  
376 .access .inner a:hover {  
377   color: #0070C0;  
378 }  
379 .access .inner a:active {  
380   color: #0056B3;  
381 }  
382 .access .inner a:visited {  
383   color: inherit;  
384 }  
385 .access .inner a:link {  
386   color: inherit;  
387 }  
388 .access .inner a:disabled {  
389   color: inherit;  
390 }  
391 .access .inner a:disabled:link {  
392   color: inherit;  
393 }  
394 .access .inner a:disabled:visited {  
395   color: inherit;  
396 }  
397 .access .inner a:disabled:active {  
398   color: inherit;  
399 }  
400 .access .inner a:disabled:hover {  
401   color: inherit;  
402 }
```

Agenda

Tech Stack

Decision Matrix

Tech Radar

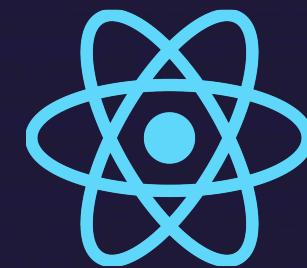
Developer Roadmaps

Tech Stack

Was ist ein Tech Stack?

Liste aller Technologien für Entwicklung und Betrieb

Daten / Persistierung



Frontend



Backend



Runtime Environment



Reality Check

Vielseitiger Tech Stack im Startup

- Learning on the job (continuously)
- Agiles arbeiten
- Migration vom Programmierer zum Produktentwickler

*Der moderne Tech Stack ist kein statisches Konzept mehr.
Entwickler müssen kontinuierlich lernen und sich anpassen.*

Tech Stack Dimensionen

Tech Stack umfasst allerdings mehr als nur die direkten Technologien

- IT Infrastructure
- Collaboration Framework
- CI/CD Pipeline
- Dokumentation
- Monitoring & Observability

Beispiel Stack

Datenbank: MongoDB

Backend: Express

Frontend: React

Runtime: Node

Wird auch MERN Stack genannt

Die zentralen Fragen

Wie komme ich zu einem Tech Stack?

Wie verwalte ich einen Tech Stack in einer Organisation?

Decision Matrix

Systematische Entscheidungsfindung

Warum eine Decision Matrix?

Wichtige technische Entscheidungen sollten:

- **Explizit** getroffen werden
- **Systematisch** ablaufen
- **Transparent** dokumentiert sein

Der 7-Schritte Prozess

1. Situation vergegenwärtigen
2. Zielzustand definieren
3. Kontext analysieren
4. Kriterien definieren
5. Alternativen finden
6. Alternativen bewerten
7. Risiken mitigieren

1. Situation vergegenwärtigen

Ziel: Gemeinsames Problemverständnis

- Problem sehr spezifisch beschreiben
- Warum ist eine Entscheidung nötig?
- Was passiert bei **keiner Entscheidung?**

2. Zielzustand definieren

Technik: Von → Nach Formulierung

Beispiel:

Von: Keine klare Tech-Strategie, blockiert Fortschritt

Nach: Definierter Tech Stack mit Plan für weitere Entscheidungen

3. Kontext analysieren

Alle **Constraints** und Einflussfaktoren verstehen:

- Technologische Einschränkungen
- Organisatorische Rahmenbedingungen
- Rechtliche/regulatorische Anforderungen
- Zeitliche Constraints
- Markt und Wettbewerb
- Stakeholder

4. Kriterien definieren

Gute Kriterien sind:

- Messbar (idealerweise objektiv)
- Umfassen
- Relevant
- Klar
- Spezifisch

Kriterien Priorisierung

Best Practice: 8-10 Kriterien

Priorisierung:

Must have Should have Could have

✗ Vage Kriterien: "Technologie soll gut sein"

5. Alternativen finden

Regeln:

- Minimum 3, Maximum 5 Alternativen
- Noch nicht bewerten (außer Machbarkeit)
- Externe Perspektiven einbeziehen
- Auch nicht-offensichtliche Optionen

6. Alternativen bewerten

Zwei-Schritt Evaluation:

1. Fakten sammeln (objektive Beschreibung)
2. Interpretation im gegebenen Kontext

Bewertungssystem

Ampel-Modell:

- Grün = erfüllt Kriterium vollständig
- Gelb = erfüllt Kriterium teilweise
- Rot = erfüllt Kriterium größtenteils nicht

Optional: Scoring mit Gewichtung

Beispiel: Alternativen bewerten

Kriterium	Priorität	NestJS + VueJS + AWS	Spring Boot + JSP + Azure
Das umsetzende Team hat genügend Erfahrung mit der Technologie	Should Have	Das Team hat in den letzten 3 Jahren 4 Projekte mit dem Tech Stack in Produktion gebraucht	Nur 2 von 5 Entwickeln haben bereits Erfahrung mit Sprint Boot. Azure ist allen bekannt.
Die Projektanforderungen können umgesetzt werden	Must Have	Umsetzbar	Umsetzbar
Die Auswahl der Technologie darf in keinem Widerspruch zu Vorgaben stehen	Must Have	Nicht Java Technologien müssen als begründete Ausnahme von einem Architekturboard genehmigt werden	Widerspricht keinen Vorgaben.

7. Risiken mitigieren

Jede Entscheidung hat Nebeneffekte

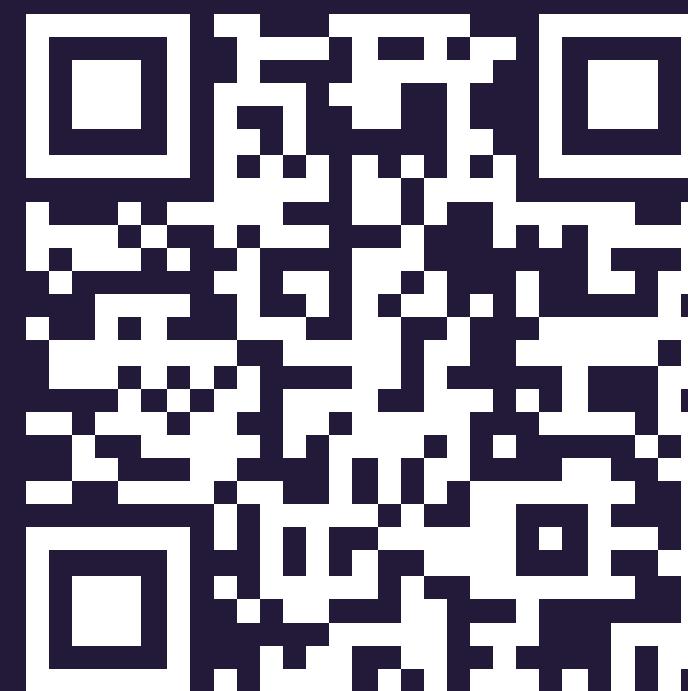
Typische Fragen:

- Welche neuen Risiken entstehen?
- Was wird schwieriger oder unmöglich?
- Wie können Risiken reduziert werden?

Template

Template mit Erklärung, Beispiel und Vorlage

<https://miro.com/app/board/uXjVLe4L0co=/>



Wann einsetzen?

- High-Impact Entscheidungen
- Unklare oder kontroverse Entscheidungen
- Situationen mit starker Bias
- Entscheidungen mit Kommunikationspflicht

Vorteile der Decision Matrix

- Reduziert kognitive und persönliche Bias
- Verbessert Entscheidungsqualität im Durchschnitt
- Schafft Transparenz und Nachvollziehbarkeit
- Fördert Team-Partizipation
- Produziert dauerhafte Dokumentation

Tech Radar

Technologie-Landkarte der Organisation

Was ist ein Tech Radar?

Visualisiert und steuert die Technologie-Landschaft

- Dokumentationstool
- Strategisches Steuerungsinstrument
- Lebendiges Artefakt

Technologie-Status

Adopt – Produktionserprobt, voll empfohlen

Trial – Erfolgreich in Use Cases, leicht höheres Risiko

Assess – Vielversprechend, noch nicht validiert

Hold – Nicht für neue Projekte empfohlen

Struktur

Segmente:

- Frameworks
- Sprachen
- DevOps
- SaaS

Essentiell: Such- und Filterfunktionen

Organisatorischer Wert

Ein Tech Radar:

- Schafft Klarheit und Alignment
- Hilft Entwicklern bei Orientierung
- Unterstützt Recruiting und Onboarding
- Leitet Sales bei Kundenakquise
- Dient als strategischer Zielzustand
- Ermöglicht geplante Migrationen

Developer Roadmaps

Karriere bewusst gestalten



Kernbotschaft

Karrierewachstum und Skill-Entwicklung sind **planbar**

Ein Roadmap ist eine **Strategie**, keine Checkliste

Wichtige Erkenntnisse

- Es gibt mehr als Web Development
- Bewusst Bereiche wählen zum Erkunden oder Ignorieren
- Lebenslanges Lernen ist eine Haltung, keine Phase
- Awareness für Disruption ist kritisch

Lernstrategie

Entwickler sollten:

- Langfristiges Ziel definieren
- Skill-Aufbau entsprechend planen
- Regelmäßig Richtung überprüfen
- Spaß daran haben

Nützliche Ressourcen

- Udemy: <https://udemy.com>
- Coursera: <https://coursera.org>
- Awesome Github Lists <https://github.com/topics/awesome>
- roadmap.sh: <https://roadmap.sh>
- TLDR Newsletter: <https://tldr.tech>