

>>> **Modernes Webdevelopment in Haskell**

>>> **Am Beispiel des Projektes mateamt**

Name: nek0 - nek0@nek0.eu

Date: 18. September 2022

## Typische Unterteilung

Typische Unterteilung

**Backend**

The diagram illustrates the typical division of modern web development into two main components: Backend and Frontend. The Backend component is represented by a gray rounded rectangle on the left, and the Frontend component is represented by a larger gray rounded rectangle on the right. Both components are labeled with their respective names in bold black text.

**Frontend**

## Typische Unterteilung

### **Backend**

Typische Sprachen:

### **Frontend**

## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP

### **Frontend**

## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP
- \* Java

### **Frontend**

## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP
- \* Java
- \* Ruby

### **Frontend**

## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP
- \* Java
- \* Ruby
- \* Python

### **Frontend**



## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP
- \* Java
- \* Ruby
- \* Python

### **Frontend**

Verwendete Sprachen:

- \* HTML

## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP
- \* Java
- \* Ruby
- \* Python

### **Frontend**

Verwendete Sprachen:

- \* HTML
- \* CSS

## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP
- \* Java
- \* Ruby
- \* Python

### **Frontend**

Verwendete Sprachen:

- \* HTML
- \* CSS
- \* JavaScript

## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP
- \* Java
- \* Ruby
- \* Python

### **Frontend**

Verwendete Sprachen:

- \* HTML
- \* CSS
- \* JavaScript

Häufig verwendete Frameworks:

## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP
- \* Java
- \* Ruby
- \* Python

### **Frontend**

Verwendete Sprachen:

- \* HTML
- \* CSS
- \* JavaScript

Häufig verwendete Frameworks:

- \* React

## Typische Unterteilung

### **Backend**

Typische Sprachen:

- \* PHP
- \* Java
- \* Ruby
- \* Python

### **Frontend**

Verwendete Sprachen:

- \* HTML
- \* CSS
- \* JavaScript

Häufig verwendete Frameworks:

- \* React
- \* Bootstrap

## Typische Unterteilung

### Backend

Typische Sprachen:

- \* PHP
- \* Java
- \* Ruby
- \* Python

### Frontend

Verwendete Sprachen:

- \* HTML
- \* CSS
- \* JavaScript

Häufig verwendete Frameworks:

- \* React
- \* Bootstrap
- \* jQuery





>>> Haskell



\* Rein funktional



- \* Rein funktional
- \* Nicht-strikt evaluiert



- \* Rein funktional
- \* Nicht-strikt evaluiert
- \* Stark statisch typisiert



- \* Rein funktional
- \* Nicht-strikt evaluiert
- \* Stark statisch typisiert
- \* Volle Typinferenz



- \* Rein funktional
- \* Nicht-strikt evaluiert
- \* Stark statisch typisiert
- \* Volle Typinferenz

\* Big Data



- \* Rein funktional
- \* Nicht-strikt evaluiert
- \* Stark statisch typisiert
- \* Volle Typinferenz
- \* Big Data
- \* Banken- und Finanzsektor



- \* Rein funktional
- \* Nicht-strikt evaluiert
- \* Stark statisch typisiert
- \* Volle Typinferenz
- \* Big Data
- \* Banken- und Finanzsektor
- \* Wissenschaft

## >>> Hauptunterschiede zu imperativer Programmierung



## >>> Hauptunterschiede zu imperativer Programmierung

- \* Strikte Trennung zwischen reinen Berechnungen und Seiteneffekten

## >>> Hauptunterschiede zu imperativer Programmierung

- \* Strikte Trennung zwischen reinen Berechnungen und Seiteneffekten
- \* Unveränderliche Variablen

## >>> Hauptunterschiede zu imperativer Programmierung

- \* Strikte Trennung zwischen reinen Berechnungen und Seiteneffekten
- \* Unveränderliche Variablen
- \* Keine Schleifen

## >>> Hauptunterschiede zu imperativer Programmierung

- \* Strikte Trennung zwischen reinen Berechnungen und Seiteneffekten
- \* Unveränderliche Variablen
- \* Keine Schleifen
- \* Kein State

## >>> Vorteile von Haskell in der Webentwicklung

## >>> Vorteile von Haskell in der Webentwicklung

- \* Serialisierung und Deserialisierung von Daten unkompliziert

## >>> Vorteile von Haskell in der Webentwicklung

- \* Serialisierung und Deserialisierung von Daten unkompliziert
- \* Datenverarbeitung kann leichter auf Korrektheit geprüft werden

## >>> Vorteile von Haskell in der Webentwicklung

- \* Serialisierung und Deserialisierung von Daten unkompliziert
- \* Datenverarbeitung kann leichter auf Korrektheit geprüft werden
- \* Webanwendungen sind in der Regel zustandslos



## >>> Vorteile von Haskell in der Webentwicklung

- \* Serialisierung und Deserialisierung von Daten unkompliziert
- \* Datenverarbeitung kann leichter auf Korrektheit geprüft werden
- \* Webanwendungen sind in der Regel zustandslos
- \* Konkurrente Anfragen/Berechnungen relativ leicht möglich

## >>> Konventionen beim mateamt im Gegensatz zum yammat

mateamt

yammat

## >>> Konventionen beim mateamt im Gegensatz zum yammat

mateamt

- \* Programmiersprache: Haskell

yammat

## >>> Konventionen beim mateamt im Gegensatz zum yammat

mateamt

- \* Programmiersprache: Haskell

yammat

- \* Programmiersprache: Haskell

## >>> Konventionen beim mateamt im Gegensatz zum yammat

mateamt

- \* Programmiersprache: Haskell
- \* Stellt nur ein Backend dar

yammat

- \* Programmiersprache: Haskell

## >>> Konventionen beim mateamt im Gegensatz zum yammat

mateamt

- \* Programmiersprache: Haskell
- \* Stellt nur ein Backend dar

yammat

- \* Programmiersprache: Haskell
- \* Front- und Backend

## >>> Konventionen beim mateamt im Gegensatz zum yammat

### mateamt

- \* Programmiersprache: Haskell
- \* Stellt nur ein Backend dar
- \* Microservice-Architektur

### yammat

- \* Programmiersprache: Haskell
- \* Front- und Backend

## >>> Konventionen beim mateamt im Gegensatz zum yammat

### mateamt

- \* Programmiersprache: Haskell
- \* Stellt nur ein Backend dar
- \* Microservice-Architektur

### yammat

- \* Programmiersprache: Haskell
- \* Front- und Backend
- \* Monolithische Architektur



# >>> Das Frontend des mateamtes

## >>> Das Frontend des mateamtes

- \* Zukünftige Benutzer des mateamtes sind ausdrücklich eingeladen eigene Frontends/Clients zu schreiben

## >>> Das Frontend des mateamtes

- \* Zukünftige Benutzer des mateamtes sind ausdrücklich eingeladen eigene Frontends/Clients zu schreiben
- \* Zur Unterstützung bei der Entwicklung wird folgendes angeboten:

## >>> Das Frontend des mateamtes

- \* Zukünftige Benutzer des mateamtes sind ausdrücklich eingeladen eigene Frontends/Clients zu schreiben
- \* Zur Unterstützung bei der Entwicklung wird folgendes angeboten:
  - \* Beschreibung der API nach OpenAPI 3.0

## >>> Das Frontend des mateamtes

- \* Zukünftige Benutzer des mateamtes sind ausdrücklich eingeladen eigene Frontends/Clients zu schreiben
- \* Zur Unterstützung bei der Entwicklung wird folgendes angeboten:
  - \* Beschreibung der API nach OpenAPI 3.0
  - \* Swagger UI

## >>> Das Frontend des mateamtes

- \* Zukünftige Benutzer des mateamtes sind ausdrücklich eingeladen eigene Frontends/Clients zu schreiben
- \* Zur Unterstützung bei der Entwicklung wird folgendes angeboten:
  - \* Beschreibung der API nach OpenAPI 3.0
  - \* Swagger UI
- \* Ein Referenzclient (der matebeamte) ist bereits in Arbeit

Bitte schnallen Sie sich an..

## >>> Code-Repositories



`gitea.nek0.eu/nek0/mateamt`



`gitea.nek0.eu/nek0/matebeamter`



Fragen? Fragen!

# Vielen Dank für die Aufmerksamkeit!

Diese Presentation steht zum Download bereit unter:



<https://gitea.nek0.eu/nek0/Webdev-Haskell>