>>> **Modernes WEbdevolpment in Haskell**
>>> **Am Beispiel des Projektes mateamt**

Name: Amedeo Molnár - nek0@nek0.eu
Date: September 7, 2022

>>> Gliederung

1. **Moderne Webentwicklung**
2. **Haskell**
   Hauptunterschiede zu imperativer Programmierung
3. **Vorteile von Haskell in der Webentwicklung**
4. **Konventionen beim mateamt im Gegensatz zum yammat**
5. **Das Frontend des mateamtes**
6. **Der Quelltext**

>>> **Moderne Webentwicklung**

>>> Moderne Webentwicklung

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

Frontend

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP

Frontend

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP
* Java

Frontend

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP
* Java
* Ruby

Frontend

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP
* Java
* Ruby
* Python

Frontend

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP
* Java
* Ruby
* Python

Frontend

Verwendete Sprachen:

* HTML

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP
* Java
* Ruby
* Python

Frontend

Verwendete Sprachen:

* HTML
* CSS

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP
* Java
* Ruby
* Python

Frontend

Verwendete Sprachen:

* HTML
* CSS
* JavaScript

Backend

Typische Sprachen:

* PHP
* Java
* Ruby
* Python

Frontend

Verwendete Sprachen:

* HTML
* CSS
* JavaScript

Häufig verwendete Frameworks:

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP
* Java
* Ruby
* Python

Frontend

Verwendete Sprachen:

* HTML
* CSS
* JavaScript

Häufig verwendete Frameworks:

* React

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP
* Java
* Ruby
* Python

Frontend

Verwendete Sprachen:

* HTML
* CSS
* JavaScript

Häufig verwendete Frameworks:

* React
* Bootstrap

>>> Moderne Webentwicklung

Backend

Typische Sprachen:

* PHP
* Java
* Ruby
* Python

Frontend

Verwendete Sprachen:

* HTML
* CSS
* JavaScript

Häufig verwendete Frameworks:

* React
* Bootstrap
* jQuery

>>> Haskell

>>> Haskell

* Rein funktional

>>> Haskell

* Rein funktional
* Nicht-strikt evaluiert

>>> Haskell

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

>>> Haskell

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

>>> Haskell

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

* Big Data

>>> Haskell

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

* Big Data
* Banken- und FInanzsektor

>>> Haskell

* 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/Berechungen relativ leicht hmöglich

>>> secname

mateamt

yammat

>>> **secname**

mateamt

* Programmiersprache: Haskell

yammat

>>> secname

mateamt

* Programmiersprache: Haskell

yammat

* Programmiersprache: Haskell

>>> secname

mateamt

* Programmiersprache: Haskell
* Stellt nur ein Backend dar

yammat

* Programmiersprache: Haskell

>>> secname

mateamt

* Programmiersprache: Haskell
* Stellt nur ein Backend dar

yammat

* Programmiersprache: Haskell
* Front- und Backend

>>> secname

mateamt

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

yammat

* Programmiersprache: Haskell
* Front- und Backend

>>> secname

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/???