Technical Report Team Gelb

1st Aaron Kreuzer
Medieninformatik
OTH Amberg-Weiden
Amberg, Deutschland
a.kreuzer@oth-aw.de

2nd Anna-Lena Gassner
Medieninformatik
OTH Amberg-Weiden
Amberg, Deutschland
a.gassner@oth-aw.de

3rd Caroline Bengart
Medieninformatik
OTH Amberg-Weiden
Amberg, Deutschland
c.bengart@oth-aw.de

4th Dieter Levin
Medieninformatik
OTH Amberg-Weiden
Amberg, Deutschland
d.levin@oth-aw.de

5th Fabian Meiler
Medieninformatik
OTH Amberg-Weiden
Amberg, Deutschland
f.meiler1@oth-aw.de

6th Harun Tacli
Medieninformatik
OTH Amberg-Weiden
Amberg, Deutschland
h.tacli@oth-aw.de

7th Patrick Pohl
Medieninformatik
OTH Amberg-Weiden
Amberg, Deutschland
p.pohl@oth-aw.de

***Abstract*—Überblick über unser Projekt, in welchem wir eine Farbgenerator-Webanwendung programmieren wollen, mit den entsprechenden Inspirationen, Anforderungen und Technologien.**

I. Problemstellung

A. Mission Statement

Ziel unseres Projekts ist die Entwicklung eines Farbgenerators. Die Bedienung und das Erstellen der Farbschemata erfolgen dabei intuitiv und ermöglichen damit sowohl Laien als auch professionellen Anwendern einen leichten Zugriff, der dadurch unterstützt wird, dass alle Funktionen auch ohne Konto nutzbar sind und das System aus einfachen Bedienungselementen besteht. Das System ist für die Entwickler und Designer ein hilfreiches Tool für die Erstellungen einer Farbpalette oder für das Finden von Inspirationen von verschiedenen Farbkombinationen. Zu den wesentlichen Features gehört die freie, einfache Erstellung von Farbpaletten, die Vorschau in Form einer Beispielwebsite und die Übersicht mit den Bewertungen und Kommentaren der einzelnen Farbpaletten. Unser Produkt unterscheidet sich von den Mitbewerbern, indem es die besten und hilfreichsten Features vereint, welche bisher immer nur einzeln vorzufinden sind.

B. Rahmenbedingungen

1. Strikte Trennung zwischen Frontend und Backend
2. Fokusierung auf Priorisierung
3. Einhalten der individuellen Deadlines
4. OS und Browserunabhängige Entwicklung muss möglich sein

II. Benutzerfreundlichkeit

A. Qualitätsziele

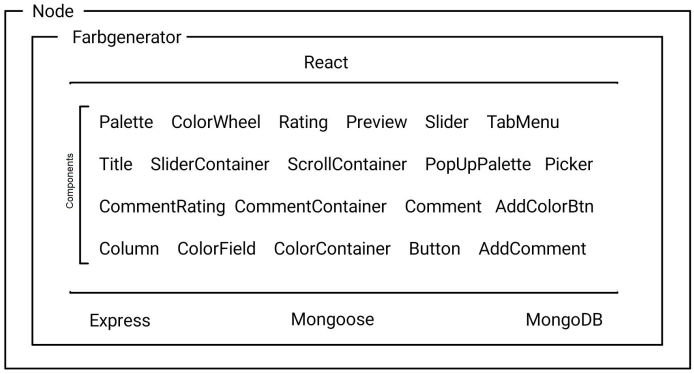
Qualitätsziel	Beschreibung
Übersichtlichkeit	Das Layout der Seite hat eine klare Struktur.
Intuitivität	Es soll ohne Vorkenntnisse einfach bedienbar sein.
Wartbarkeit	Strikte Trennung und klare Benennung der Komponenten. Einzelne Komponenten können verändert bzw. entfernt werden ohne Auswirkungen auf das System. Verwendung gleicher Programmierkonventionen
Zuverlässigkeit	Das System steht den Anwendern jederzeit zur Verfügung.

B. Größte Risiken

- Updates von Nodemodulen und die dadurch möglichen Veränderungen von Strukturen
- Verlust der Datenbank durch Fehler und Abstürze

III. Lösungsstrategie

A. Informelles Überblicksbild



B. Lösungsstrategie

Qualitätsziel	Lösungsansätze
Übersichtlichkeit	- Intuitives, modernes Web-Userinterface - Anwendung der Gestaltgesetze
Intuitivität	- Verwendung von bekannten Benutzerelementen - Sofortige Rückmeldung auf Benutzereingaben
Wartbarkeit	- Komponentenbasierte Entwicklung - Einsatz gängiger JavaScript-Open-Source-Lösungen - Paketverwaltung mit npm
Zuverlässigkeit	- Node-Applikationsserver - MongoDB Sharding für horizontales Scaling - Regelmäßige Backups der Datenbank

C. Architekturprinzipien

- Bevorzuge vorhandene Module anstatt eigene zu schreiben
- Halte dich an die Programmierkonventionen
- Jede Komponente hat ihren eigenen Code und CSS

IV. Lösungsdetails

A. Architekturentscheidung

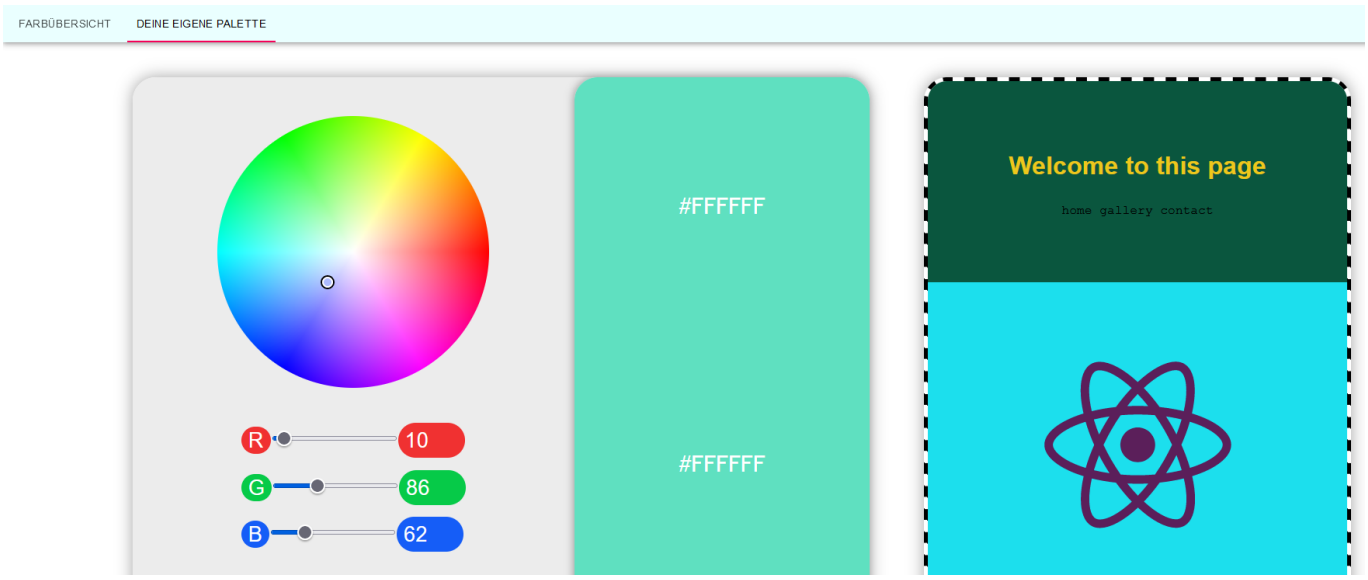
Für die Technologie zur Implementierung einer Webapplikation gibt es mehrere potentielle Möglichkeiten, wobei wir uns für den MERN-Stack entschieden haben. Gleichwertige Alternativen wären der MEVN- oder der MEAN-Stack gewesen. Diese Stacks bestehen jeweils fest aus MongoDB, für die Datenbankseite, Express.js und Node.js, für die Serverseite, und entweder aus React.js (MERN), Angular (MEAN) oder Vue.js (MEVN) für die clientseitige Applikation. Vorteil aller drei Stack Varianten ist, dass alle verwendeten Technologien in JavaScript programmierbar sind, bzw. mit JSON-Dokumenten arbeiten. Für uns als Entwickler hat es den Vorteil, dass Daten nicht erst aufwändig in andere Formate umgewandelt werden müssen, bevor sie verwendet werden können. Auch, dass nur auf eine Programmiersprache zurückgegriffen wird, die beherrscht werden muss, ist ein klarer Vorteil. Damit ist der ME(R/A/V)N-Stack, der Technologie-Stack der Wahl für Webentwickler, die dynamische Web-Interfaces mit hohem JSON Anteil programmieren wollen, was exakt auf unser Projekt zutrifft. Da alle drei Stacks sich bei der Datenbank und den serverseitigen Applikationen gleichen, fiel die Wahl zwischen MERN, MEVN oder MEAN also vor allem wegen der Clientseitig verwendeten Technologie. Bei Angular und Vue.js handelt es sich um TypeScript bzw. JavaScript Webframeworks, während React eine JavaScript-Library darstellt. Auch wenn die Begriffe des Frameworks und der Library häufig synonym verwendet werden, gibt es Unterschiede zwischen beiden. So hat der Entwickler bei einer Library die volle Kontrolle und kann sie seinen eigenen Wünschen entsprechend erweitern, anpassen und hat die Kontrolle über den Ablauf. Bei einem Framework liegt die Kontrolle über den Ablauf bei dem Framework, zwar kann auch hier der Entwickler seinen eigenen Code an vorgesehenen Stellen einfügen, aber das Framework entscheidet, wann es diesen aufruft. Ausschlaggebend sich für React zu entscheiden war neben der Tatsache, dass es dem Entwickler die volle Kontrolle bietet, auch dass es etwas leichter zu erlernen ist als Angular oder Vue.js und leicht um weitere Module, wie z.B. Material-UI, ergänzt werden kann. Alternative Möglichkeiten zu den bereits beschriebenen Technologien wären zum Beispiel: WISA, Ionic, MySQL, Spring o.ä. gewesen. WISA, ein Technologie-Stack von Windows, kam deshalb nicht in Frage, da für die Nutzung Lizenzen erworben werden müssen und sich dieser Stack eher für sehr anspruchsvolle und komplexe Webprojekte anbietet. Bei Ionic handelt es sich um eine Client-Seitige Anwendung, die vor allem für die Entwicklung von plattformübergreifenden nativen oder auch hybriden Apps entwickelt wurde, da wir aber eine reine Web-Applikation schreiben wollten, wäre dies für unsere Zwecke zu viel gewesen. Spring ist eine auf Java beruhende Technologie die z.B. das Schreiben von REST-Apis ermöglicht, da es hier aber mit Express.js schon eine Alternative gibt, die auch direkt in JavaScript programmiert werden kann, bot sich für diese Zwecke Express.js eher an als Spring. MySQL wäre eine Alternative für die Datenbank gewesen und hätte durch zusätzliches Laden von Modulen auch in Verbindung mit Node.js und Express.js verwendet werden können, allerdings kann MySQL nicht direkt mit JSON-Dokumenten arbeiten, weshalb ein entsprechendes Umwandeln und Verarbeiten der Daten nötig gewesen wäre. Da dieser Schritt bei MongoDB entfällt, bot sich hier MongoDB eher an als MySQL.

B. Technologie-Stack

- JavaScript
- Node
- Express
- MongoDB
- React
- HTML
- CSS

C. Abbildung: Erstellte Paletten auf der Startseite

D. Abbildung: Ausschnitt Seite "Deine eigene Palette"



E. Quellen

- https://www.mongodb.com/mern-stack
- https://www.ionos.de/digitalguide/server/knowhow/web-stack-grundlagenwissen-und-beispiele/
- https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/
- https://www.ionos.de/digitalguide/server/knowhow/web-stack-grundlagenwissen-und-beispiele/
- https://material-ui.com/components/tabs/
- https://casesandberg.github.io/react-color/
- https://styled-components.com/docs/basics
- https://github.com/jaames/iro.js
- https://iro.js.org/guide.html#color-picker-options
- https://iro.js.org/advanced.html#available-components