Hochschule Luzern – Informatik

Bachelorarbeit Herbstsemester 2022
Software-Entwicklung, Human Computer Interaction Design

Entwicklung einer Webseite für die Präsentation und Kommunikation der Inhalte der UX Research Gruppe der HSLU-Informatik

---

Autor

Jan Kalbermatter

Betreuungsperson

Marcel Uhr

Auftraggeber

HSLU-Informatik
Immersive Realities Center
UX Research Group

Experte

Martin Burri

Abgabedatum: 06. Januar 2023

Bachelorarbeit an der Hochschule Luzern – Informatik

Titel: Entwicklung einer Webseite für die Präsentation und Kommunikation der Inhalte der UX Research Group der HSLU-Informatik

Studentin/Student: Jan Kalbermatter

Studiengang: BSc Informatik

Abschlussjahr: 2023

Betreuungsperson: Marcel Uhr

Expertin/Experte: Martin Burri

Auftraggeberin/Auftraggeber: HSLU-Informatik – Immersive Realities Center, UX Research Group

Codierung / Klassifizierung der Arbeit:
☒ Öffentlich (Normalfall)
☐ Vertraulich

Eidesstattliche Erklärung
Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe angefertigt habe, alle verwendeten Quellen, Literatur und andere Hilfsmittel angegeben habe, wörtlich oder inhaltlich entnommene Stellen als solche kenntlich gemacht habe das Vertraulichkeitsinteresse des Auftraggebers wahren und die Urheberrechtsbestimmungen der Hochschule Luzern respektieren werde.

Ort / Datum, Unterschrift

_____

Abgabe der Arbeit auf der Portfolio Datenbank:
Bestätigungsvisum Studentin/Student
Ich bestätige, dass ich die Bachelorarbeit korrekt gemäss Merkblatt auf der Portfolio Datenbank ablege. Die Verantwortlichkeit sowie die Berechtigungen gebe ich ab, so dass ich keine Änderungen mehr vornehmen oder weitere Dateien hochladen kann.

Ort / Datum, Unterschrift

_____

Verdankung

An dieser Stellte möchte ich mich bei allen Personen bedanken, welche am Projekt beteiligt waren und mich dabei unterstützt haben.
Zuerst möchte ich meinem Betreuer Marcel Uhr für Unterstützung bei der Durchführung bedanken. Weiter möchte ich mich bei den Mitarbeitenden der UX Research Group für ein äusserst spannendes Projekt und einer interessanten und konstruktiven Zusammenarbeit danken. Abschliessend möchte ich mich auch noch bei Aleksa Zivadinovic, Carina Vasconcelos Duarte und Annika Hovingh für das Feedback, Durchführen der Tests und die Interviews bedanken. Die Zusammenarbeit mit den beteiligten Personen und deren Unterstützung habe die Umsetzung dieses Projektes erleichtert.

Abstract

Die UX Research Group der HSLU hat keinen eigenen Web-Auftritt sondern lediglich eine Unterseite auf der Offiziellen Webseite der HSLU. Es besteht aus Seiten der Forschungsgruppe das Bedürfnis, die eigenen Kompetenzen zu vermitteln, die Zusammenarbeit anzubieten und ihre Wissen zu vermitteln.
Im Rahmen einer vorherigen Bachelorarbeit wurde ein eigener Design-Prozess entworfen und mit einem Figma-Prototypen visualisiert.

Ziel der vorliegenden Arbeit ist es einen Web-Auftritt für die UX Research Group anhand diese Prototypen zu erstellen. Die Vision der Forschungsgruppe bezüglich der Webseite muss abgeklärt werden und der Prototyp wird mit dem entsprechenden Feedback erweitert. Der Kernteil der Umsetzung bildet das erstellte Empfehlungs-System, welches es Benutzer erlaubt passende Methoden für ihre eigenen Projekte zu finden. Weitere Features wie das Erstellen von Forschungsprojekten, Events oder Blogeinträgen wurden im umgesetzt. Während der Durchführung des Projektes wurden in mehreren Workshop und Interview Feedback zur momentanen Umsetzung aufgenommen. Am Schluss des Projekts wurde die Funktion und Usability mittels Test validiert.

Dieses Dokument befasst sich mit der Erarbeitung der Webseite. Es soll aufzeigen, welche Schritte zum Erfüllen der Anforderungen nötig waren und welche Schwierigkeiten dabei aufkamen.

Inhaltsverzeichnis

1 Problem, Fragestellung, Vision

1.1 Problem

Die UX Research Group der Hochschule Luzern hat keinen eigenen Webauftritt und ihre Möglichkeiten sich zu präsentieren und ihre eigenen Dienste anzubieten sind auf der offiziellen HSLU-Webseite eingeschränkter, als sie es mit einer eigenen Lösung wären. Im Rahmen einer Bachelorarbeit im Herbstsemester 2021 wurde von Nico Iseli eine Erweiterung des Human-Centred-Design (HCD) Vorgehens nach ISO 9241-210 erstellt. Zudem wurde mit Figma[1] ein Prototyp[2] für einen Webauftritt erstellt, welcher unter anderem diesen Prozess abbildet.

1.2 Aufgabenstellung

Das Ziel dieser Arbeit ist es, das von Nico Iseli erstellte Konzept in einer Webseite zu konkretisieren, erweitern und umzusetzen.
Bezüglich der Erweiterung des Konzepts sollen Workshops gemeinsam mit der UX Research Group der Hochschule Luzern durchgeführt werden, in wessen Rahmen die Vision der Forschungsgruppe konkretisiert werden soll. Resultieren aus diesen Workshops soll der Figma Prototype und die aufgenommen Vision in einer Web-Applikation umgesetzt werden. Vor der Durchführung dieser Umsetzung, sollen verschiedene Tools analysiert werden und gemeinsam mit der Forschungsgruppe wird ein Entscheid gemäss des Technologie Stacks gefällt. Das Hosting der Webseite soll abgeklärt werden und die Webseite wird nach Projektdurchführung mit einer gemeinsam gewählten Domain veröffentlicht. Die Web-Applikation soll durch einige Community-Funktionen, wie zum Beispiel Blogbeiträge und Events, erweitert werden. Eine erste Umsetzung des Prototyps wird mittels Workshops validiert und das Feedback wird in die Dokumentation aufgenommen oder direkt umgesetzt. Die fertige Web-Applikation wird mittels *Usability Tests* und *Blackbox Tests* auf ihre Funktionalität und User Experience getestet. Zudem soll darauf geachtet werden, dass der Inhalt der Webseite responsiv an die Bildschirmgrösse angepasst wird und sowohl auf dem Desktop, einem Tablet und auch Mobiltelefonen ansprechend aussieht.
Die komplette *Aufgabenstellung* kann im Anhang gefunden werden

1.3 Anforderungen

Die Anforderungen werden aus der Aufgabenstellung und aus den Workshops mit der UX Research Group abgeleitet. Diese Anforderungen werden in Nicht-Funktionale und Funktionale Anforderungen aufgeteilt und deren Validation kann in den jeweiligen Kapiteln gefunden werden. Die *Anforderungen* sind als separates Dokument im Anhang zu finden.

[1] https://www.figma.com/files/recent?fuid=970297727377141709
[2] https://www.figma.com/proto/Ec020p0preWDK3h3cQbSNI/Prototype?node-id=1%3A4&starting-point-node-id=1%3A4

1.3.1 Funktionale Anforderungen

Die Funktionale Anforderungen an das Projekt werden folgend aufgelistet:

Tabelle 1: Funktionale Anforderungen

#	Anforderung	Validierung
Inhalt		
1.1	Das Empfehlungs-System wurde umgesetzt	Mittels Blackbox Test
1.2	Der HCD-Prozess wird beschrieben	Auf Webseite vorhanden. In Dokumentation wird auf relevante Kapitel verwiesen.
1.3	Die Methoden aus Nico Iselis Vorarbeit wurden übernommen	Auf Webseite vorhanden. In Dokumentation wird auf relevante Kapitel verwiesen.
1.4	Die Kontaktaufnahme mit der Forschungsgruppe ist möglich	Auf Webseite vorhanden. In Dokumentation wird auf relevante Kapitel verwiesen.
1.5	Die Kompetenzen und Forschungsgruppe selbst werden vorgestellt	Auf Webseite vorhanden. In Dokumentation wird auf relevante Kapitel verwiesen.
CMS/Backend		
2.1	Neue Seiten können selbst erstellt und mit Inhalt befühlt werden	Im CMS Handbuch ersichtlich
2.2	Der Inhalt der Seiten ist versioniert im Backend zu finden	Im CMS Handbuch ersichtlich

1.3.2 Nicht Funktionale Anforderungen

Die Nicht-Funktionale Anforderungen an das Projekt werden folgenden aufgelistet:

Tabelle 2: Nicht-Funktionale Anforderungen

#	Anforderung	Validierung
Performance		
1.1	Akzeptable Performance auf Desktop und Mobile	Validierung mit Lighthouse. Optimale wäre ein Score > 90. 80 > ist als akzeptable zu sehen.
1.2	Akzeptable Accessibility auf Desktop und Mobile	Validierung mit Lighthouse. Optimale wäre ein Score > 90. 80 > ist als akzeptable zu sehen.
1.3	Akzeptables SEO der Webseite	Validierung mit Lighthouse. Optimale wäre ein Score > 80. 70 > ist als akzeptable zu sehen.
Benutzerfreundlichkeit		
2.1	Benutzer finden den gewünschten Inhalt einfach	Usability wird mittels User-Tests validiert

2.2	Benutzer können die Webseite unabhängig von Bildschirmgrösse benutzen	Responsive-First. Validierung auf Desktop, Tablet und Mobile
2.3	Die Webseite sieht für den Benutzer ansprechend aus	Validierung mittels Interviews und User-Tests
Dokumentation		
3.1	Entscheiden gemäss CMS und Frontend sind nachvollziehbar dokumentiert	In der Auswertung der Kriterien wird auf die jeweiligen Stellen in der Dokumentation hingewiesen
3.2	Ein kurzes Handbuch, welches den Einsatz des CMS beschreibt, ist vorhanden.	In der Auswertung der Kriterien wird auf das Dokument hingewiesen. Erstellen von neuen Inhalten und Seiten im CMS muss klar beschrieben werden

2 Stand der Forschung

2.1 Prototyp

Für die Umsetzung der Arbeit liegt ein Prototyp vor, welche in einer früheren Bachelorarbeit erstellt wurde. Der Prototyp wurde mithilfe des Tools Figma erstellt und kann unter der folgenden URL gefunden werden: Prototyp – Holistic Design

Der Prototyp soll als Basis für die Umsetzung und für das Design der Webseite dienen. Die Umsetzbarkeit der einzelnen Features des Prototypen muss bei der Auswahl des Content Management Systems (CMS) beachtet werden.

2.2 Content Management System

Für die Umsetzung des Prototypen soll ein CMS eingesetzt werden, damit es für die Mitarbeitenden der UX Research Groups auch in Zukunft möglich ist, einfach Inhalt und neue Seiten zu erstellen.
Ein Content Management System ist eine Software, welche es ermöglicht Content aufzunehmen, zu bearbeiten oder zu löschen. Es gibt verschiedene Arten von CMS wie beispielsweise Enterprise Content Management Systems (ECM). Dieses dient zur Verwaltung und Organisation von Dokumenten eines Businesses und zur Gewährleistung der Sicherheit der Daten und dass diese an die korrekten Benutzer geliefert wird (Mitarbeitenden, Kunden, Stakeholder). Relevant für diese Projekt sind Web Content Management Systems (WCMS) welche es Benutzer erlaubt, Webseite auch ohne Wissen von Web-Development selber zu erstellen und neuen Inhalt aufzunehmen.[3] Für die Umsetzung des Projektes wurden zwei verschiedene Arten von WCMS (folglich nur CMS genannt) in Betracht bezogen

2.2.1 All-In-One CMS

Die erste Art von CMS, sind die welche man als «All-In-One»-CMS bezeichnen könnte. Das heisst die ganze Webseite wird vom CMS aufgebaut. Backend und Frontend des CMS hängen fest zusammen. Das Frontend wird vom CMS selbst erstellt und mit dem aufgenommenen Inhalt aus dem Backend befüllt. Oftmals wird ein "What You See Is What You Get" (WYSIWYG) Editor zur Verfügung gestellt, über welchen die Benutzer ihre Webseite visuell aufbauen können
Viele All-In-One CMS bieten dem Entwickler auch die Möglichkeit, die Vorlagen für die Erstellung des Frontend anzupassen und diese dadurch zu individualisieren. Für weniger technisch versierte Benutzer existiert hier oftmals die Möglichkeit von anderen Benutzern erstellte Plugins und Themes (Designs) für den eigenen Web-Auftritt zu kaufen oder sogar gratis zu übernehmen.
Als Beispiel hierfür kann WordPress genommen werden. WordPress ist die beliebteste CMS-Lösung und gut 40% aller Webseite im Internet basieren darauf[4]. Durch diese weite Verbreitung von WordPress gibt es ein riesiges Netzwerk and WordPress-Entwicklern, welche ihre eigenen Designs und Plugins zur Verfügung

[3] https://www.ixiasoft.com/types-of-content-management-systems/
[4] https://w3techs.com/technologies/details/cm-wordpress

stellen[5]. WordPress bietet auch eine Decoupled-Funktionalität an, welche es ermöglicht Inhalt über eine API einem Frontend zur Verfügung zu stellen.

2.2.2 Headless CMS

Die zweite Art von CMS sind sogenannte Headless CMS. Ein Headless CMS stellt dem Benutzer, genau gleich wie ein All-In-One CMS, ein Interface zur Verfügung über welches Inhalt verwaltet werden kann. Zusätzlich zu diesem Interface wird von Headless CMS jedoch noch ein Application Programming Interface (API)[6] zur Verfügung gestellt an welche Entwickler Anfragen senden und den CMS-Inhalt abfragen können.
Einen WYSIWG Editor, wie bei den All-In-One CMS, ist in den meisten Headless CMS nicht möglich da Headless CMS auch mehrere Frontends mit verschiedenen Designs mit Inhalt bedienen müssen.
So könnten zum Beispiel neben einer Webseite auch noch eine IOS sowie eine Android App mit Inhalt befüllt werden. Headless CMS bieten dem Entwickler einer Webseite mehr Flexibilität, indem die Entscheidung über den Frontend Technologie Stack und Architektur komplett frei gelassen wird.
Das Ändern dieses gewählten Frontend Stacks ist auch einfacher als bei einem All-In-One CMS, da dieses nicht mit den gespeicherten Daten zusammenhängt und daher das CMS genau gleich beibehalten werden kann.
Da das Frontend und Backend nicht zusammenhängen, kann das ganze System auch als stabiler angesehen werden, da ein Absturz des Backend nicht zugleich bedeutet das dem Benutzer im Frontend ein Error angezeigt wird.
Je nach Wahl des Technologie Stacks können auch Rendering Methoden wie Server-Side-Rendering (SSR) oder Static Side Generation (SSG) (Siehe *NextJS*) eingesetzt werden, um die Webseite noch performanter zu machen. (Zanas, 2022)

2.2.3 Usability-Tests

Die Usability-Test sollen eingesetzt werden, um die Handhabung der Webseite durch den Endbenutzer zu validieren und weiteres Feedback für die Weiterentwicklung zu erhalten. Es hilft dem Designer vor allem folgende Ziele zu erreichen:

- **Probleme Identifizieren:** Probleme im Design oder sogar im Produkt selbst
- **Chancen aufdecken:** Chancen für Verbesserungen
- **Den Endbenutzer kennenlernen:** Präferenzen und Verhalten
-

Es gibt verschiedene Arten von Usability Tests, wie zum Beispiel das Hallway-Testing, wobei Passanten befragt werden oder den Labor Test, wobei die Beobachtung im Vordergrund steht[7]. Gemeinsamkeiten der meisten Usability-Tests bestehen aus den folgenden drei Punkt:

1. **Moderator:** Hilft den Teilnehmern durch den Testprozess
2. **Aufgaben:** Gibt die Aktivitäten des Teilnehmers vor

[5] https://wordpress.org/plugins/
[6] https://en.wikipedia.org/wiki/API
[7] https://humancentreddesign.ch/methods/usability-testing

3. **Teilnehmer:** Ein realistischer Benutzer des zu studierenden Produkts

Als Moderator soll einerseits der Teilnehmer interviewt werden, andererseits soll dessen Verhalten mit dem Produkt beobachtet und notiert werden. Wichtig ist es, beim Interview das Verhalten des Teilnehmers nicht zu beeinflussen und so die Testresultate zu verfälschen (Moran, 2019).

2.2.4 Black-Box Testing

Während eines Black-Box Tests wird die Funktionalität eines Systems getestet, ohne das das sich die Testperson über die Implementierung des Systems auskennt. Das getestete System befindet sich hier effektiv in einer Black-Box, durch die der Benutzer nicht hindurchsehen kann, er kann lediglich damit interagieren und das entsprechende Resultat seiner Interaktion sehen.

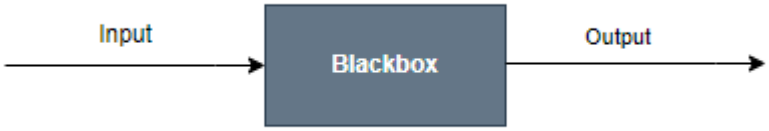


Abbildung 1: Blackbox

Da sich der Entwickler eines Systems meist mit dem fertigen System gut auskennt, ist es möglich, dass dieser unbewusst Annahmen zum System trifft und beim Testen nicht ganz nach Spezifikation vorgeht. Als Testpersonen sollten entsprechend Personen gewählt werden, welche sich nicht vertieft mit dem System auskennen.

Ziel des Black-Box Testing ist es, die Spezifikationen eines Systems zu überprüfen. Ausgehend von diesen Spezifikationen werden Testfälle definiert, welche sicherstellen sollen, dass der vorgegebene Funktionsumfang eingehalten wird. Je nach Spezifikationen und vorhandener Zeit, ist es auch nicht immer wirtschaftlich, die ganzen Spezifikationen ausgiebig zu testen, sondern sich auch die wesentlichen Funktionen des Systems zu fokussieren. (Wikipedia, 2022)

Die drei prominentesten Arten des Black-Box Testing sind die folgenden:

- **Funktionales Testing:** Funktionalität und Features eines Systems überprüfen
- **Nicht-Funktionales Testing:** Nicht-Funktionale Anforderungen wie die Performance oder Usability eines Systems
- **Regressions Testing:** Testen ob Änderungen am System (z.B. am Code) andere Funktionalitäten beeinflusst hat.

Der Ablauf eines Black-Box Test kann in die folgenden Schritte gegliedert werden:

- Die Voraussetzungen und Spezifikationen des Systems werden analysiert
- Es werden die zu testenden Spezifikationen ausgewählt
- Es werden jeweils valide und invalide Eingaben für die Tests definiert
- Die Soll-Outputs der Tests werden definiert
- Die Tests werden von einer Testperson genau nach Definition durchgelaufen
- Testresultate werden mit den Soll-Outputs verglichen

- Fehler im System werden behoben

Dieser Ablauf kann iterativ durchlaufen werden, um behobene Fehler oder neue Features immer auf ihre Funktionalität zu validieren.

Diese Art von Tests können auch mit Tools automatisiert werden. Mit dem Tool Selenium[8] können zum Beispiel Browser simuliert werden. Dadurch können die User-Inputs von funktionalen Black-Box Tests, wie zum Beispiel das Eingeben eines Textes in ein Input-Feld und das Klicken eines Buttons, simuliert werden. Die in der Web-Applikation angezeigten Resultate können anschliessend eingelesen und validiert werden.[9] Auch Nicht-Funktionale Black-Box Tests, wie zum Beispiel Performance Tests oder Load Tests, können mit Tools wie LoadRunner[10] automatisiert werden. (Hamilton, 2022)

2.2.5 NextJS

NextJS ist ein Framework, welches die Entwicklung von Web-Applikationen vereinfachen soll. NextJS setzt das beliebte JavaScript Library ReactJS ein, um das User-Interface zu rendern. ReactJS wird von Meta (ehemalig Facebook) entwickelt. Laut einer Umfrage von StackOverflow ist React bei gut 44% der Befragten professionell angestellten Entwicklern im Einsatz. NextJS selbst wird von 14% der Befragten eingesetzt [11].
NextJS als Framework nimmt dem Entwickler einige Sachen ab, welche er in ReactJS selbst verwalten müsste. Server-Side Rendering (SSR), Routing und Bilder-Optimierung werden von NextJS verwaltet.

Routing in NextJS wird zum Beispiel über einen Ordner in der Dateistruktur des Projekts namens ***pages*** verwaltet. Alle daran existierenden Ordner selbst sind auch Pfade. Eine Datei index.js kann erstellt werden, welche immer am Stamm dieses Pfads angezeigt wird. Pfade können auch dynamisch definiert werden. Eine Ordnerstruktur wie ***pages/blog/[slug].js*** kann benutzt werden, um dynamisch für jeden Blogeintrag einen eigenen Pfad zu erstellen. Für einen Blogeintrag namens „Hello World“ wäre dieser Pfad ***pages/blog/hello-world.***

NextJS bietet mehrere Möglichkeiten an, Daten von einem Server abzufragen und einzelne Seiten darzustellen. So können Seiten mit Server-Side Rendering oder mittels Static Site Generation (SSG) pre-renderd werden. Bei der Static Site Generation werden für alle Dateien im ***pages*** Ordern während des Build eine HTML-Datei generiert. Der Endnutzer erhält beim Besuchen der Webseite vom Server nicht die vom Entwickler erstellten JavaScript-Dateien, sondern die während des Build anhand dieser generierten HTML-Dateien.
Anders dazu kann auch Client Side Rendering (CSR) eingesetzt werden. Hierbei werden die Daten für das Befüllen der Webseite erst abgefragt, nach dem der Benutzer die Webseite geöffnet hat. Dadurch muss mehr JavaScript geladen werden und es kann vorkommen, dass Inhalt auf der Webseite nicht von Anfang an angezeigt und erst nach einigen Sekunden geladen wird (NextJS, 2022).

[8] https://www.selenium.dev/
[9] https://www.selenium.dev/documentation/overview/details/
[10] https://www.microfocus.com/en-us/products/loadrunner-professional/overview
[11] https://survey.stackoverflow.co/2022/#most-popular-technologies-webframe-prof

Auf Konzepte, welche für die Umsetzung des Projekts eingesetzt wurden, wird im Kapitel *NextJS* näher eingegangen.

2.2.6 Konkurrenz

Um mögliche Ideen zu sammeln und zu sehen, welche Features von anderen Web-Applikationen von UX-Forschungsgruppen angeboten werden, werden solche Web-Applikationen gesucht und genauer analysiert.
Die Recherche ergab die folgenden Web-Applikationen:

- User Interface Design (Webseite)
- Uid.com (Webseite)
- Universität Bern (Webseite)
- MMI Basel (Webseite)

User Interface Design ist eine allgemeine Wissens-Sammlung rund um Digitales Product Design, Human Centred Design und Technologie Trends. Verschiedene Design Themenfelder wie Human Centred Design, User Experience Design oder Design Methoden können vom Benutzer näher angeschaut werden. Unter diesen Themen können jeweils Blogbeiträge gefunden werden, welche zum Thema passen. Das Design der Webseite ist sehr modern und kleine Animationen beim über Elemente Hovern, im Hintergrund der Seite und beim Scrollen lassen die Seite sehr dynamisch wirken.

Uid.com die Webseite eines Business, welches UX-Design, User Research, UX-Strategie und Software Engineering anbietet.

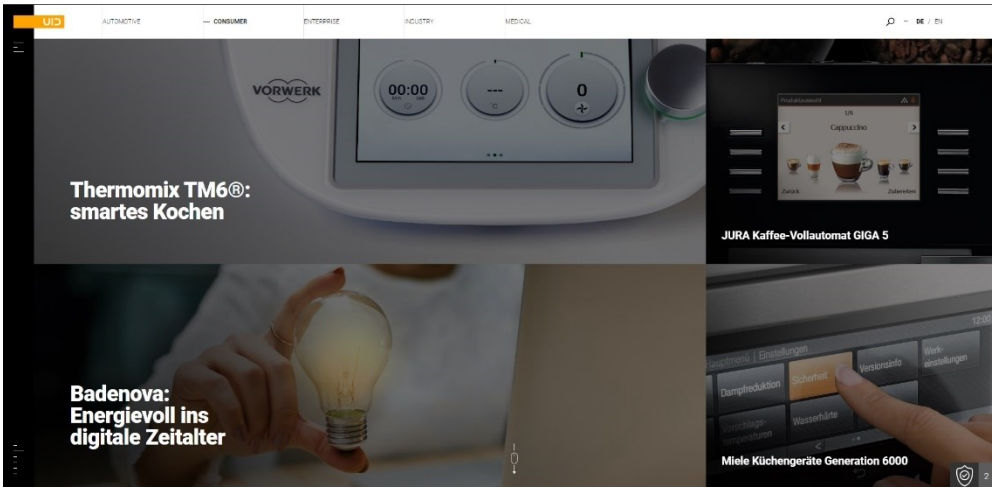


Abbildung 2: uid.com - Projekte

Durchgeführte Projekte werden in einer Bilderlandschaft (siehe Abbildung 2) dargestellt. Spezifisches UX-Wissen wurde auf der Webseite nicht vermittelt, sondern es werden lediglich die eigenen Kompetenzen vermittelt (durch die eigenen Projekte, so wie auch dem Design der Webseite selbst) und die Zusammenarbeit angeboten.

Die Webseite wurde mit sehr viele Animationen und Bildern aufgebaut, was den Inhalt sehr lebhaft wirken lässt. Ein Mangel, der beim Bedienen der Webseite direkt aufgefallen ist, ist dass das Scrolling mit dem Mausrad des Benutzers von

der Webseite „überschrieben“ wird. Das sogenannte Scroll-hijacking wird als ein Usability-Albtraum beschrieben und im Falle der analysierten Webseite wurde diese Interaktion als sehr verwirrend und unangenehmen wahrgenommen.

Der Webauftritt des User Experience Teams der **Universität Bern** ist separat von der eigentlichen Webseite der Universität[12]. Der Inhalt der Webseite ist wird auf nur drei Tabs gegliedert.

- **Home**: Hier befindet sich eine Kurze Übersicht über den Zweck der Webseite, einige Referenzen des Teams (durchgeführte Projekte), eine kurze Vorstellung der Mitglieder des Teams sowie eine erste Verlinkung auf das Kontaktformular
- **Referenzen**: Durchgeführte Projekte des Teams. Die Projekte werden alle untereinander aufgelistet und es gibt keine Möglichkeit diese zu filtern. Zuunterst auf der Seite wird ein weiteres Mal auf den Kontakt verwiesen
- **Kontakt**: Ein Kontaktformular, mit welchem eine Offerte beim Team angefragt werden kann. In einer Sidebar sind zudem noch die Kontaktdaten und ein Standort aufgelistet.

Das Design dieser Webseite ist im Vergleich mit den vorderen beiden viel simpler gehalten und wirkt auf den ersten Blick wie ein Templates Design eines CMS und nicht wie eine eigene Lösung. Diese Annahme konnte bestätigt werden indem der Source der Webseite untersucht wurde (mittels Entwickler-Konsole, F12). In dieser Source sind die beiden Ordner wp-content und wp-includes zu finden, welche von WordPress generiert werden.[13]

Das Forschungsteam für Mensch – Maschinen Interaktion der Universität Basel (**MMI Basel**) legt das Hauptmerkmal der Webseite auf ihre Forschungsprojekte. Direkt beim Besuch der Webseite erscheint ein Header, welcher auf diese verweist. Besuche werden mit Cards dargestellt und werden vor allem mit Text-Inhalt beschrieben. Alle Mitglieder des Teams werden vorgestellt und jedes Teammitglied hat zudem eine eigene Seite, auf welcher seine Forschungsinteressen, Auszeichnungen und Publikationen dargestellt werden.

Die Zusammenarbeit mit dem Team und offene Stellen werden auf der Webseite auch beworben. Zudem können potenzielle Studierende auch noch Informationen zum Studium erhalten.

Auch das Design dieser Webseite ist simpel gehalten und erinnert an das Design der Universität Bern.

Im Rahmen des Projekts gilt es, ein gutes Mittelmass zu finden, um den eigenen Inhalt übersichtlich darzustellen und Informationen einfach zu vermitteln. Das Design soll dabei trotzdem noch lebhaft wirken und der Benutzer soll wissen, wann er mit Elementen auf der Webseite interagieren kann. Der Figma-Prototyp[14]

[12] https://www.unibe.ch/
[13] https://www.elegantthemes.com/blog/resources/how-to-tell-if-a-website-is-made-with-wordpress
[14] https://www.figma.com/proto/Ec020p0preWDK3h3cQbSNI/Prototype?node-id=1%3A4&starting-point-node-id=1%3A4

bildet hierzu bereits eine gute Vorlage durch den Einsatz von interaktiven Bild-Elementen und der gewählten Akzentfarbe (Magenta).

3 Ideen und Konzepte

Für die Umsetzung des Projekts wurden verschiedene Ideen und Konzepte evaluiert. Dieses Kapitel soll eine Übersicht über die verschiedenen Lösungsansätze geben. Zudem wird der Entscheid des Technologie-Stacks genauer begründet.

3.1 CMS Auswahl

In einem ersten Anlauf wurden drei verschiedene CMS-Lösungen ausprobiert. Dazu wurde jeweils ein Account vom Autor erstellt und den Mitarbeitenden der UX Research Group zur Verfügung gestellt. Das Setup, welches für diese Test vorgenommen wurde, war minimal und die Mitarbeitenden durften eigenständig die CMS-Lösung anschauen. Ihnen wurde ein Dokument mitgegeben, welches die verschiedenen Logins und Gedanken zu der jeweiligen Lösung seitens Autors festhalten. Das Dokument ist im Anhang unter *Erste Runde* zu finden.

In diesem ersten Anlauf wurden Lösungen von WordPress, Webflow und Prismic evaluiert (siehe *Erste Runde*). In einem zweiten Anlauf wurden Prismic, zusammen mit NextJS sowie OctoberCMS evaluiert (siehe *Validierung und zweite Runde*). Der eigentliche Auswahl Prozess, welche zu der Entscheidung geführt hat, Prismic zusammen mit NextJS einzusetzen wird im Kapitel *Auswahl* beschrieben

Die verschiedenen CMS-Lösungen wurden jeweils, falls vorhanden, in einem Docker-Container vom Autor selbst gehostet und über eine Sub-Domain zu Testzwecken den Mitarbeitenden der UX Research Group zur Verfügung gestellt. Bei CMS-Lösungen, welche nicht selbst gehostet werden können, wurde ein Login erstellt, welches an die Mitarbeitenden der UX Research Group weitergegeben wurde.

3.2 NextJS

3.2.1 Static Site Generation

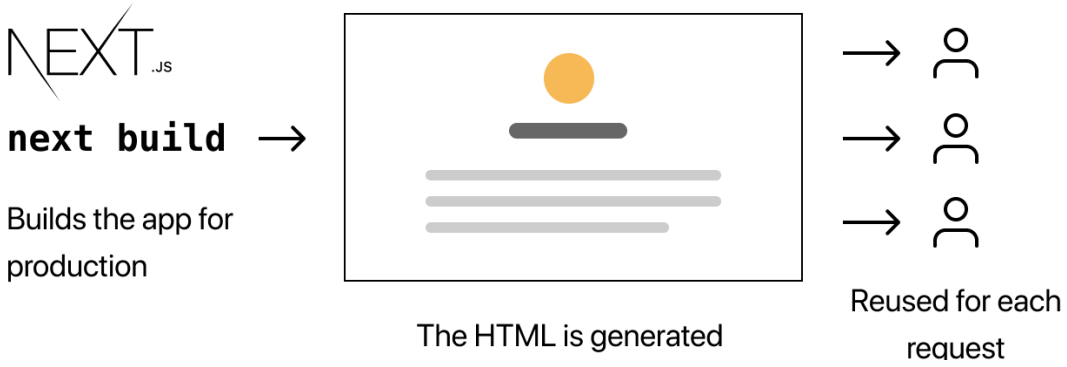


Abbildung 3: NextJS Static Site Generation

Das Konzept des Static-Site Generation wird im Projekt mithilfe von NextJS eingesetzt, um die Seiten jeweils während der Build-Time zu generieren. Dadurch kann sich der Endkunde Ladezeiten und der Entwickler und Verwalter der Seite Bandbreite und somit auch Hosting-Kosten sparen. Wird die Methode `getStaticProps` von einer Seite exportiert, wird diese automatisch von NextJS während dem Build generiert.
Laut NextJS offizieller Dokumentation wird es empfohlen `getStaticProps` zu benutzen, falls einer oder mehrere der folgenden Punkte erfüllt werden:

- Die Daten, welche für das Rendern der Seite benötigt werden, sind bereits während dem Build verfügbar
- Die Daten kommen aus einem Headless CMS
- Die Seite muss pre-rendered werden und soll schnell geladen werden
- Die Daten können öffentlich gecached werden. Keine Benutzer-spezifischen Daten müssen angezeigt werden.

Durch diese Punkte ist klar, dass sich SSG besonders gut für unseren Anwendungsfall eignet. Alle Daten sind für den Build vorhanden und durch den Einsatz von Webhooks werden durch Änderungen an Dokumenten auch immer automatisch neue Builds ausgelöst. Das eingesetzte CMS Prismic ist zudem auch ein Headless CMS.
Das pre-rendering eignet sich vor allem für Search Engine Optimization (SEO). In den *Anforderungen* wurde erwähnt, dass ein akzeptabler SEO-Score erreicht werden soll. Das schnelle Laden der Webseite ist von Seiten der UX her ein muss, damit Benutzer nicht wegen langen Ladezeiten ihr Interesse verlieren und von der Web-Applikation frustriert werden
Die Daten auf der Seite können alle öffentlich gecached werden, da es keine Benutzer-Accounts gibt. Es müssen keine nutzerspezifischen Daten auf der Webseite angezeigt werden, da jedem Benutzer dieselbe Webseite angezeigt wird.
GetStaticProps wird immer auf dem Server, und nie auf dem Client, beziehungsweise im Browser des Benutzers, ausgeführt. Wird mit dem ***next build*** Kommando ein Build ausgeführt, so wird ***getStaticsProps*** ausgeführt, um die Daten der Webseite zu holen. ***GetStaticProps*** kann mittels ISR auch im

Hintergrund den Inhalt der Seite neu validieren und anpassen indem der Funktion das ***revalidate*** Property gesetzt wird. Der Wert dieser Properties sollte ein Integer sein, welcher die minimale Zeit angibt, welche gewartet werden muss bevor eine Seite neu generiert wird. Dadurch löst der Benutzer nicht bei jedem Reload der Webseite eine Regeneration aus, sondern nur in angemessenen Abständen.
(NextJS, 2022)

3.2.2 Hooks

NextJS basiert auf ReactJS, weshalb die in ReactJS vorhandenen Hooks auch hier verwendet werden. Hooks helfen dem Entwickler unteranderem den State (Zustand) von Komponenten zu verwalten. Die Motivation von ReactJS auf das Konzept der Hooks umzusteigen, war vor allem die Wiederverwendbarkeit und geminderte Komplexität, die dadurch ermöglicht wird.[15] Die selben Features welche Hooks ermöglichen, waren zuvor bereits in Klassen umsetzbar[16]. Die neue Art mit Hooks spart hierbei jedoch Arbeit.

Die beiden wichtigsten Hooks sind ***useState*** und ***useEffect***. Es existieren noch weitere Hooks, diese waren für die Umsetzung im Projekt jedoch nicht von sehr grosser Bedeutung. Eine Auflistung der mögliche Hooks kann Hier gefunden werden.

3.2.2.1 useState

Die ***useState*** Hook ermöglicht es, die Daten einer Komponente zwischen re-renders zu persistieren. Ein re-rendern der Seite wird dann ausgelöst, wenn sich die Daten auf der Seite ändern und daher das Interface erneuert werden muss. Hierbei kann es sich um Daten aus einem API-Request handeln, oder aber ein interaktives Element welches durch eine Benutzerinteraktion verändert wird.

```
import React, { useState } from 'react';

function useStateExample() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Abbildung 4: Verwendung von useState

[15] https://reactjs.org/docs/hooks-intro.html#motivation
[16] https://reactjs.org/docs/hooks-state.html#equivalent-class-example

Der Code-Ausschnitt in Abbildung 4 zeigt, wie ***useState*** verwendet werden kann, um einen Zähler zu kreieren.
Der Zustand des Zähler wird mit `const [count, setCount] = useState(0)` als die Zahl 0 definiert. Die Funktion ***useState*** gibt zwei Werte zurück, einerseits den Parameter, welcher im Beispiel auf 0 gesetzt wird. Zudem wird eine Funktion zurückgegeben, welche es erlaubt den Wert dieses Zustands zu ändern und einen re-render auszulösen.
Klickt der Benutzer auf den Button, so wird dessen onClick -Event ausgelöst, welches wiederum die von ***useState*** zur Verfügung gestellte Methode ***setCount*** aufruft und den aktuellen Wert von ***count*** um eins erhöht. (React, 2019)

3.2.2.2 *useEffect*

Die ***useEffect*** Hook kann dafür eingesetzt werden, Seiteneffekte in ReactJS auszuführen. Als Seiteneffekte sind in ReactJS zum Beispiel GET-Request, direkte Manipulation des Document Object Models (DOM) oder der Einsatz von Timer-Funktionen wie ***setTimeout()*** in JavaScript zu verstehen. Das Re-Rendering von Komponenten in ReactJS wird von ReactJS selbst kontrolliert und der Benutzer kann einen Re-Render einer Komponente nicht verhindern. Werden Operationen mit Seiteffekten direkt in einer Komponente ausgeführt kann das zu einer Endlosschleife von Re-Renders sorgen.
Die ***useEffect*** Hook erlaubt es nun hier, das Rendering von den Seiteneffekten zu trennen und dieses Problem der Endlosschleife zu umgehen.

```
useEffect(() => {
    document.title = `Greetings to ${name}`;
  }, [name]);
```

Abbildung 5: Verwendung useEffect

Im Beispiel (siehe Abbildung 5) ist zu sehen wie die ***useEffect*** Hook eingesetzt werden kann um das DOM einer Webseite anzupassen.
UseEffect nimmt hierbei zwei Parameter entgegen. Ein Callback und ein Array mit Dependencies, welches jedoch optional ist. Beim Callback handelt es sich um eine Funktion, in welcher die Seiteneffekte ausgeführt werden sollten.
Mittels des Dependency Array kann das Verwalten von useEffect folgendermassen angepasst werden:

- **Kein Parameter:** UseEffect wird nach jedem Render der Komponente ausgeführt
- **Leeres Array (`[]`):** UseEffect wird nur **ein einziges Mal**, und zwar beim ersten Rendern der Komponenten, ausgeführt.
- **Array mir Props oder States (`[prop, state, …]`):** Die Properties der Komponente selbst und die States, welche von dieser verwaltet , können als Parameter angegeben werden. ***UseEffect*** wird beim ersten Rendern ausgeführt und danach nur, falls einer der Werte im Dependency Array ändert.

(Pavlutin, 2022)

Falls useEffect eine Funktion zurückgibt, wird diese beim nächsten Rendern der Komponente vor dem Ausführen von useEffect noch aufgerufen. Das könnte zum Beispiel zu Einsatz kommen, falls in einer Komponente zum Beispiel ein Online-Status oder ein News-Feed abonniert und ständig eingelesen wird. Damit auch nur eine Verbindung zu diesen beobachteten Werten besteht, muss im useEffect eine Cleanup-Methode dafür sorgen, dass die Verbindung vor dem Erstellen einer Neuen immer getrennt wird.[17]

[17] https://reactjs.org/docs/hooks-effect.html#example-using-hooks-1

3.2.3 Hydration

Hydration ist eine Technik, welche für das SSR und SSG von NextJS automatisch eingesetzt wird. Durch Hydration werden die statischen Seiten, welche vom Server gesendet werden zu dynamischen Seiten gemacht, indem alle Event-Handler zu den HTML-Elementen hinzugefügt werden. Diese Technik hat ihre Vor- und Nachteile
Einerseits ist der sogenannte «first contentful paint» sehr schnell, das heisst der Endbenutzer bekommt schon direkt Inhalt auf der Seite angezeigt. Andererseits handelt es sich hierbei lediglich um Inhalt welche noch nicht «hydriert» wurde und daher nicht interaktiv ist, auch wenn es für den Benutzer so aussehen mag. (Wikipedia, 2022)

Mehrere Dinge können zu Problemen bei der Hydration führen:

- Invalides HTML
    - Ein -Element in einem <p>-Element
- Code welche auf Dinge verlässt welche zwischen dem Pre-rendering und dem Browser unterschiedlich sind
    - Die globale Browser-Variable ***window*** kann für Probleme sorgen, dass diese Browserspezifisch ist und auf dem Server während dem Build nicht existiert.
- Externe Libraries
    - Oftmals mit externen CSS Libraries, hierbei müssen bei NextJS einige Dinge beachtet werden (z.B. durch Anpassen der Konfiguration)

(NextJS, 2022)

3.2.4 TailwindCSS

Die TailwindCSS Library erleichtert die Umsetzung eines Responsive Design um einiges. TailwindCSS war bereits im getesteten Starterkit implementiert und die Developer-Experience mit Tailwind wurde als sehr angenehm befunden, weshalb das CSS der Web-Applikation auch mithilfe von TailwindCSS erstellt wurde.
Der Vorteil von TailwindCSS ist, dass kein eigener CSS Code geschrieben werden muss, sondern es existieren entsprechende Klassen, welche den CSS-Code zur Verfügung stellen. Zum Beispiel kann die Klasse ***flex*** einem HTML-Element hinzugefügt werden. TailwindCSS sorgt dann entsprechend dafür das dieses Element automatisch das ***display:flex*** CSS-Property erhält.
TailwindCSS ist nach einer Mobile-First Mentalität aufgebaut, weshalb Komponenten welche damit designed werden auch schon direkt responsiv sind

3.3 Prismic

Prismic[18] ist ein Headless CMS. Sektionen einer Seite bestehen bei Prismic nicht aus statischen Teilen, sondern aus wiederverwendbaren Blöcken (genannt Slices). Diese Slices können lokal mithilfe der *Slicemachine* erstellt werden. Eine Voraussetzung für die Verwendung der Slicemachine ist, dass für das Frontend entweder das NextJS oder NuxtJS Framework eingesetzt werden. Mit der Slicemachine können automatisch Datenmodelle für eigene Dokument-Typen und Slices erstellt werden. Dokument-Typen sind Daten, zum Beispiel eine Seite oder ein Blogbeitrag, welche im CMS aufgenommen werden können. Diese Dokument-Typen und Slices können aus der lokal laufenden Slicemachine Instanz in das CMS importiert und dort verwendet werden. Es ist auch möglich Dokument-Typen (keine Slices) direkt im CMS zu erstellen, dieser Weg Dokument-Typen aufzunehmen wird jedoch von Prismic selbst als veraltet bezeichnet und es wird empfohlen, Slicemachine einzusetzen. (Prismic, 2022)

Prismic bietet die Möglichkeit Releases zu erstellen. Dokumente können zu diesen Releases hinzugefügt und anschliessend gemeinsam an einem geplanten Zeitpunkt veröffentlich werden. Benutzer haben auch die Möglichkeit, ältere Versionen von Dokumenten zu vergleichen und falls gewünscht wiederherzustellen. Veraltete Beiträge können per Knopfdruck archiviert werden. Die Benutzung des CMS wurde im *CMS-Handbuch* genauer beschrieben.

3.3.1 Slicemachine

Slicemachine vereinfacht die Arbeit mit Prismic und ermöglich es, schnell neue Komponenten und Dokument-Typen zu erstellen. Bei der Erstellung von Dokument-Typen kann entweder ein «Single Type» oder ein «Repeatable Type» erstellt werden. Single Types eignen sich für einzigartige Seiten wie die Homepage oder eine Datenschutzseite. Repeatable Types eignen sich für Dokumente von welchen mehrere erstellt werden sollen. Beispiele dafür sind Blogbeiträge, Personen oder Produkte. Zu jedem Typ kann eine Auswahl an Datenfeldern, wie ein Bild, ein Link, ein Rich-Text Editor oder eine Verbindung zu anderen Dokumenten hinzugefügt werden. Zudem können erstellte Slices für die einzelnen

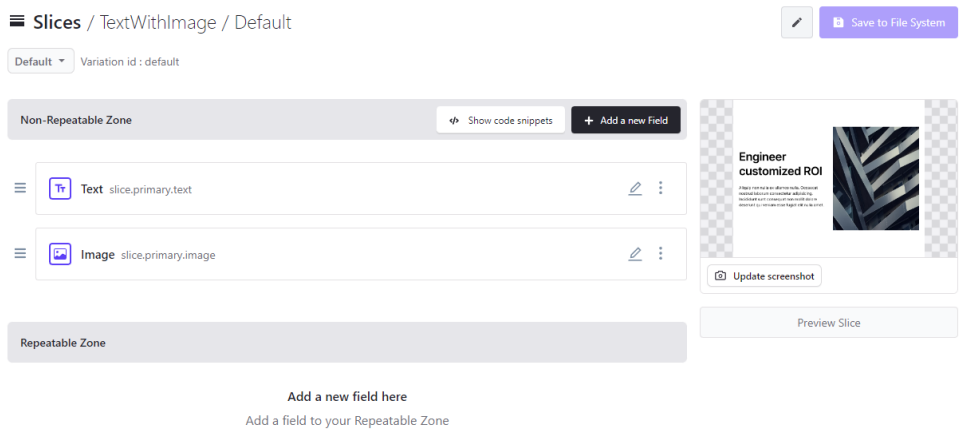


Abbildung 6: TextWithImage Slice in Slicemachine

[18] https://prismic.io/

Dokument-Typen freigeschalten werden. So können auch Slices spezifisch für einen Dokument-Typen erstellt werden.

In der Abbildung 6 zu sehen ist eine mit Slicemachine erstellte Komponente. Hier können Felder hinzugefügt werden, welche nur einmal ausgefüllt werden können sowie weitere Felder welche beliebig oft wiederholt werden können. Slicemachine bietet auch die Funktion an, die Komponenten während der Entwicklung live anzusehen. Die Slices werden dazu von Slicemachine automatisch mit Mock-Daten gefüllt. Vor dem Hochladen der Slices in das Prismic CMS kann ein Screenshot entweder automatisch erzeugt oder manuell hinzugefügt werden.

3.4 Wireframes

Das Empfehlungs-System bildet einen Kernteil des Projekts und der Fokus bei der Umsetzung soll am Anfang hierauf liegen. Im Rahmen des Workshops mit der UX Research Group wurden einige Skizzen zur potenziellen Darstellung dieses Empfehlungs-Systems erstellt. Anhand dieser ersten Skizzen wurden mit Balsamiq Wireframes erstellt. Es wurde zudem auch bereits einige Wireframes für die potenzielle Darstellung der Forschungsprojekte und die Darstellung von Methoden erstellt. Das Balsamiq-File mit allen erstellten Wireframes ist im Anhang unter *Wireframes* zu finden.

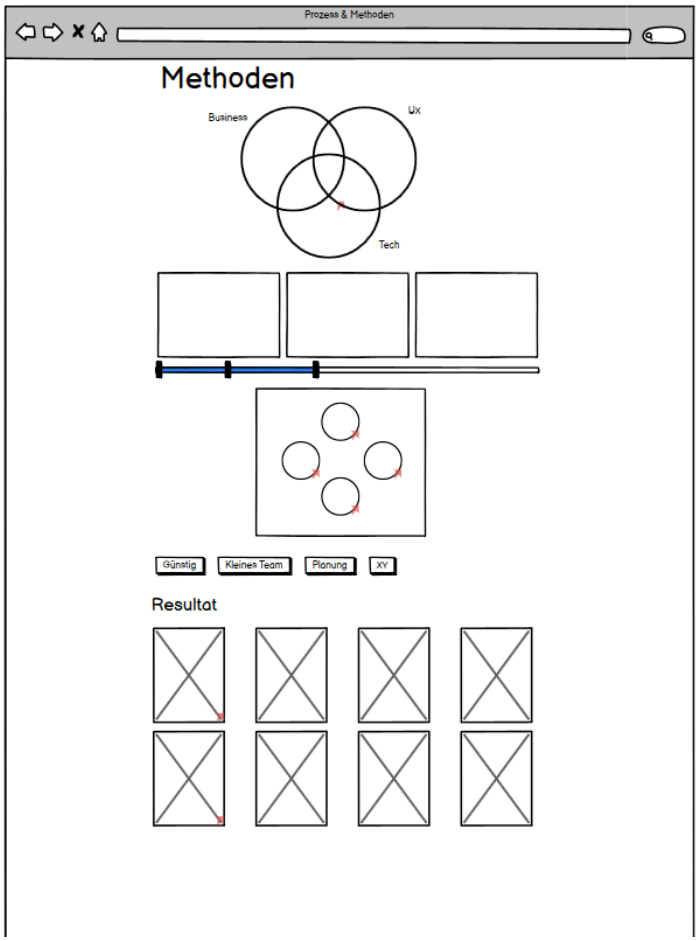


Abbildung 7: Wireframe - Empfehlungs-System

Im Wireframe (Abbildung 7) zu sehen sind die fünf verschiedenen Blöcken, welche auch in der finalen Version das *Empfehlungs-System* ausmachen. Das Venn-Diagramm mit den darunter stehenden drei Blöcke und dem Slider, mit welchen die momentane Phase des Projekts gewählt werden kann. Die Prozess-Engine, welche ein Modal öffnet um verschiedene Methoden für die Forschung, Definition, Design oder Validierung zu finden und die Methoden-Filter, welche bereits im Figma-Prototypen[19] im Teil des Empfehlungs-Systems beinhaltet waren.

Auf dem Wireframe (Abbildung 8) ist die Darstellung der Forschungsprojekte zu sehen. Zuunterst auf der Seite ist noch ein Button vorhanden, welcher den Benutzer zu den im Projekt benutzten Prozessen und Methoden führen würde. Aufgrund von Feedback aus der UX Research Group werden die Methoden eines Projekts mithilfe der Projekt-*Engine* Komponente dargestellt, welche auch in Abbildung 7 zu sehen ist.
Ein wichtiger Punkt ist es, dass Forschungsprojekte auch mit Bild und Video-Elementen dokumentiert werden können, um den Inhalt möglichst verständlich und interessant zu vermitteln.

Abbildung 8: Wireframe - Forschungsprojekt

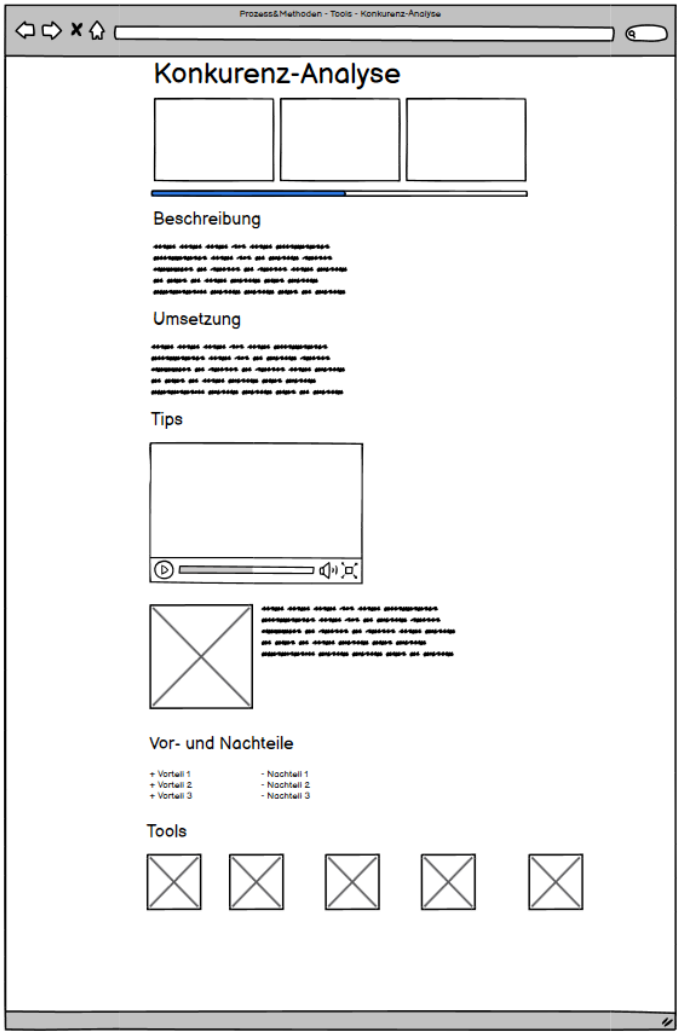


Abbildung 9: Wireframe - Methode

Der Wireframe (Abbildung 9) zeigt die Ansicht einer Methode. Ziel dieser Seite ist es, eine Methode und deren Umsetzung dem Benutzer zu erklären. Auch hier ist wiederum wichtig, dass die Beschreibung einer Methode mit Video und Bild gestaltet werden kann. Zuoberst einer jeden Methode war geplant, immer den *Status*-Block anzuzeigen, um schnell einzuordnen, wo eine Methode passt. Die Umsetzung erfolgt jedoch näher dem Beispiel aus dem Figma-Prototypen, wobei der Prozessschritt (Planung, Forschung, Definition, Design, Validation, Finalisierung) immer im Titel ersichtlich ist

4 Methoden

4.1 Projektmethode & Organisation

4.1.1 Projektmethode

Das Projekt wird nach der Methode SODA geführt

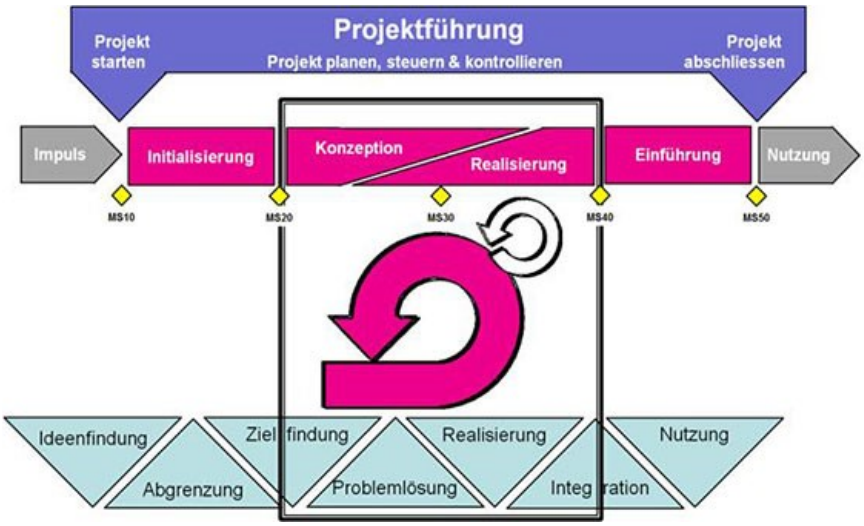


Abbildung 10: SODA Ablauf

SODA kombiniert die klassische und agile Projektführung miteinander. Diese Projektmethode eignet sich besonders gut für mittelgrosse Softwareprojekte und ermöglicht es durch mehrere Sprints während des Entwicklungsprozesses flexibel zu bleiben und gut auf wechselnde Anforderungen zu reagieren.

4.1.2 Organisation

In der folgenden Tabelle sind die Stakeholder in diesem Projekt zu finden

Name/Bezeichnung	Rollen
UX Research Group	Auftraggeber
Uhr Marcel	Betreuer
Burri Martin	Experte
Kalbermatter Jan	Projektleiter, SCRUM Master, Product-Owner, SCRUM Team

Nachfolgend die Aufgaben der Projektrollen:

- **Projektleiter:** Rahmenplanung, Meilensteindefinition, Organisation
- **SCRUM Master:** Qualitätssicherung
- **Product-Owner:** Sprint-Planing, Priorisierung
- **SCRUM Team:** Entwicklung

4.1.3 Rahmenplan

Nachfolgend ist eine übersichtliche Version des Rahmenplans zu finden mit den jeweiligen Meilensteinen. Eine ausführliche Version des Rahmenplans kann im Anhang unter *Rahmenplan* gefunden werden.

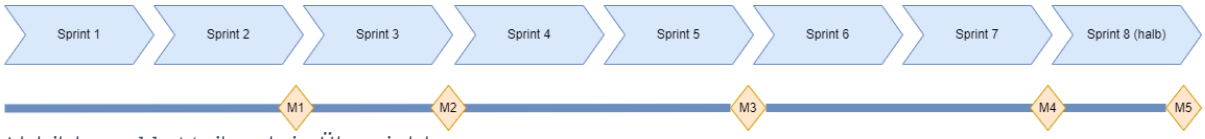


Abbildung 11: Meilenstein Übersicht

Tabelle 3: Meilenstein Beschreibung

Meilenstein	Beschreibung	Datum
M1	Entscheidung gemäss CMS und Technologie Stack getroffen.	
M2	Begin der technischen Umsetzung und Abschluss Konzeptionsphase.	
M3	Erste Version der Technischen Umsetzung abgeschlossen.	
M4	Web-Applikation validiert, entsprechend Feedback angepasst und eingeführt.	
M5	Abgabe	

4.1.4 Projektphasen

Das Projekt kann in die folgenden vier Phasen eingeteilt werden:

Phase	Start	Ende	Sprints
Initialisierungsphase	19.09.2022	03.10.2022	1
Konzeptionsphase	03.10.2022	31.10.2022	2
Realisierungsphase	01.11.2022	18.12.2022	3.5
Einführungsphase	19.12.2022	25.12.2022	1.5

4.1.5 Sprintplanung

Das Projekt wird in zweiwöchige Sprints eingeteilt. Am Ende eines jeden Sprints wird der vergangene Sprint evaluiert und die Arbeiten für den nächsten Sprint werden geplant. Die Sprint-Review und Sprintplanungen können im Anhang der Arbeit unter *Sprintplanung* gefunden werden

4.1.6 Risikomanagement

Bei Projektbeginn wurde ein Risikomanagement erstellt, inklusive zwei Risikomatrizen und Massnahmen. Bei Beginn jedes Sprints wurden Risiken evaluiert und falls vorhanden inklusive Massnahmen im Sprintplanungs-Dokument aufgenommen. Jedes Sprint-Dokument enthält die aktuellen Risiken für den Sprint. Diese sind immer als zusätzliche Risken zu den anfangs definierten Risiken zu sehen.

Das initiale Risikomanagement inklusive der Risikomatrizen kann im Anhang unter *Risikomanagement* gefunden werden und ist zudem unten aufgelistet. Auch zu sehen sind zwei Risikomatrizen, welche diese initialen Risiken analysieren.

Nr.	Risikobeschreibung	Eintrittwahrscheinlichkeit	Schadensausmass
Projektumfang			
1	Unzureichend definierter Projektumfang	3	3
2	Ungenaue Schätzung der benötigten Zeit	3	3
3	Unerwartete Anpassungen	4	2
4	Ungenau definierte Änderungen	3	2
Auftraggeber			
5	Auftraggeber hat ungenaue/falsche Erwartungen	3	3
Ressourcen			
6	Zeitmangel für die Umsetzung der Arbeit	3	3
7	Leistungsmangel der Projektarbeitenden	2	3
Technisches			
8	Technische Umsetzung stellt sich als zu komplex heraus	3	4
9	Technische Umsetzung der Anforderung stellt sich als nicht möglich heraus	3	3
10	Die Software erfüllt ihre funktionalen Anforderungen nicht	4	3
11	Die Software erfüllt ihre nicht-funktionalen Anforderungen nicht	4	2
Anforderungen			
12	Anforderungen sind mehrdeutig oder unklar formuliert	4	3
13	Anforderungen sind unvollständig	3	2
Software			
14	Schlechte Programmierung	4	2
15	Software funktioniert unzuverlässig	5	3

Tabelle 4: Risikomanagement

Die daraus erstellte Risikomatrix sieht wie folgt aus:

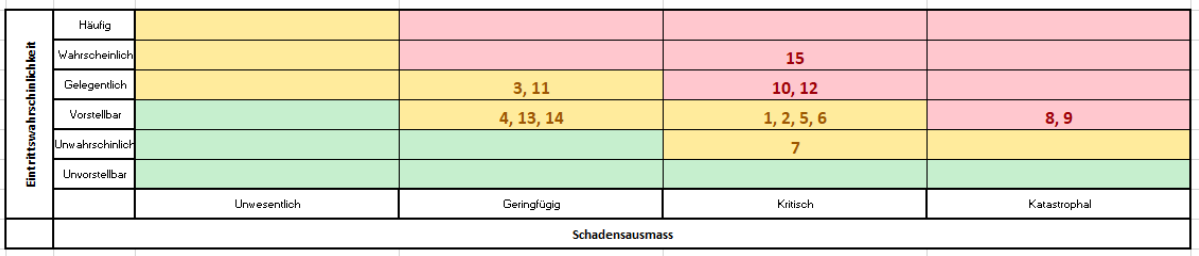


Abbildung 12: Risikomatrix - keine Massnahmen

Zu den entsprechenden Risiken wurden jeweils die folgenden Massnahmen definiert:

Tabelle 5: Massnahmen

Nr.	Massnahme	Eintrittswahrscheinlichkeit nach Massnahme	Schadensausmass nach Massnahme
Projektumfang			
1	Genaue Rücksprache mit dem Auftraggeber halten. Genaue Überlegungen zu Zeitaufwand und Machbarkeit machen.	1	2
2	Wichtige Features priorisieren. Rahmenplan erstellen.	2	1
3	Zusammenarbeit mit Auftraggeber zum Definieren der zu erreichenden Ziele. So früh wie möglich	3	2
4	Genügend Zeit aufwenden, um Änderungen sauber zu planen. Bei Unklarheiten direkt Nachfragen.	2	2
Auftraggeber			
5	Gemeinsam mit Auftraggeber Rahmen des Projekts klarstellen	2	2
Ressourcen			
6	Arbeiten/Zeit vor Beginn der Arbeit einplanen. Zeitplan laufend evaluieren und falls nötig mehr Zeit aufwenden.	1	2
7	Wöchentlich Arbeiten einplanen.	1	2
Technisches			
8	Ausreichende Recherche vor der Programmierung. Dokumentation zur Hilfe nehmen.	2	3
9	Ausreichende Recherche vor der Programmierung bezüglich des Technologie Stacks und dessen Möglichkeiten.	2	3
10	Testen der Funktionalität im Rahmen der der Sprint-Reviews. Abschliessende Blackbox Tests.	3	2
11	Evaluierung und Anpassungen. Einsatz eines Technologie Stack welche Performance und Accessibility vereinfacht.	3	2
Anforderungen			
12	Rücksprache mit Auftraggeber	2	2
13	Anforderungen gemeinsam mit Auftraggeber definieren.	2	2

Software			
14	Clean Code Prinzipien beachten. Wiederverwendbaren Code schreiben. Bisherige Erfahrung aus Studium und Beruf einsetzen	2	2
15	Laufendes Testen der Funktionalität. Abschliessendes Testing vor der Abgabe.	2	1

Nach dem Definieren der Massnahmen ist eine klare Verbesserung der Risikomatrix zu sehen:

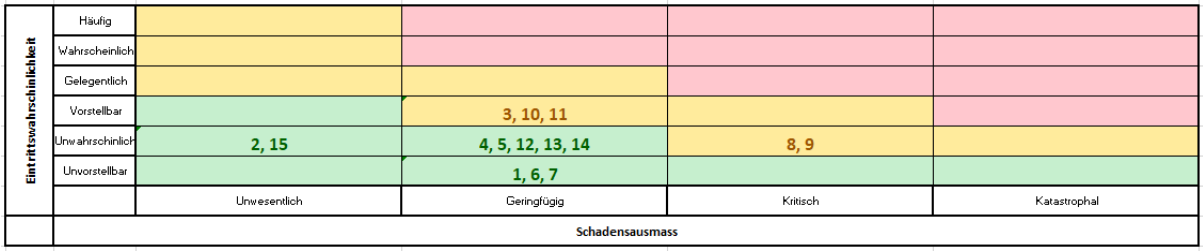


Abbildung 13: Risikomatrix - mit Massnahmen

Die fünf Risiken, welche besonders herausstachen, waren:

- **8:** Technische Umsetzung stellt sich als zu komplex heraus
- **9:** Technische Umsetzung der Anforderung stellt sich als nicht möglich heraus
- **11:** Die Software erfüllt ihre funktionalen Anforderungen nicht
- **12:** Anforderungen sind mehrdeutig oder unklar formuliert
- **15:** Software funktioniert unzuverlässig

Das Ausmass der Risiken 8, 9, 11, und 12 lässt sich mehrheitlich durch eine gute Planung bei der Auswahl des Technologie Stacks und Rücksprache mit dem Auftraggeber deutlich mindern. Das Risiko 15 verlangt eine Teststrategie und die konsequente Durchführung dieser.

4.1.7 Teststrategie

Die Funktionalität der Web-Applikation soll mittels *Blackbox Tests* sichergestellt werden. Nach Abschluss der Entwicklung wurde die Funktionalität der gesamten Webseite auf Fehler getestet. Die Tests (siehe Anhang) werden von drei Personen (abgesehen vom Autor) durchgeführt. Die Testprotokolle dazu können jeweils im Anhang unter *Blackbox-Tests* gefunden werden.

Die Usability der Webseite ist ein wichtiges Thema und soll im Rahmen des Projekts auch getestet werden. Diese Tests werden erst gegen Schluss der Entwicklungsarbeit durchgeführt. Einerseits werden Workshops durchgeführt, in welchen mit Mitarbeitenden der Forschungsgruppe oder mit potenziellen Benutzern die Webseite gemeinsam analysiert wird. Zudem werden auch noch *Usability Tests* mit potenziellen Benutzern durchgeführt. Im Rahmen dieser Usability Tests wurden Aufträge für die User definiert. Das Lösen dieser Aufgaben wurde beobachtet und es wurden einige Fragen dazu gestellt, welche mehr Aufschluss darüber geben sollen, wie einfach beziehungsweise schwierig einem die Bedienung der Webseite fällt. Die Protokolle der Usability-Test können im Anhang unter *Usability-Tests* gefunden werden.

5 Realisierung

Dieses Kapitel befasst sich mit der Umsetzung der zuvor erwähnten Ideen und Konzepten.

5.1 Workshop – Vision der UX Research Group

Um das Projekt zu starten, wurden zusammen mit der UX Research Group ein Workshop geplant. Dieser Workshop diente dazu, die Vision der Mitarbeitenden der Forschungsgruppe für die Webseite festzuhalten. Die im Rahmen des Workshops aufgezeichneten Beispiele und Grafiken können als Bilder im Anhang unter *Workshop - Bilder* gefunden werden.

5.1.1 Resultat

In einem ersten Workshop wurden die folgenden Resultate erzielt welche relevant für die Umsetzung der Web-Applikation sind/

5.1.1.1 Inhalt

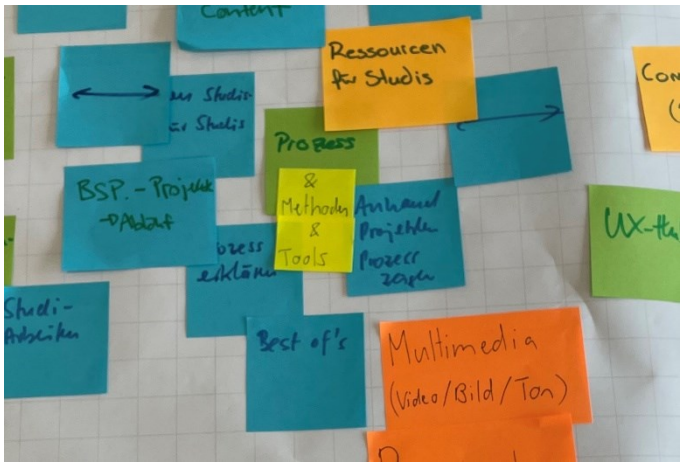


Abbildung 14: Ausschnitt aus dem Brainstorming

Der Inhalt wurde gemeinsam mit der Forschungsgruppe definiert. Zusammen mit den Mitgliedern der UX Research Group wurde ein Brainstorming durchgeführt und die verschiedenen Wünsche und Anforderungen der Mitarbeitenden aufgenommen.

Aus dem Brainstorming kamen mehrere Inhaltspunkte heraus, welche als besonders wichtig zu beachten sind, da sie von mehreren Mitgliedern erwähnt oder in der anschliessenden Diskussion als solche definiert wurden. Die wohl wichtigsten Inhalte sind die folgenden:

- **Forschungsprojekte:** Eigene Forschungsprojekte einfach aufnehmen und veröffentlichen
- **Kompetenzen:** Die Kompetenzen der UX Research Group vermitteln. Werbung in eigener Sache machen.
- **Module/Major:** Informationen zu den Modulen und dem Major vermitteln.
- **Community:** Interaktion mit den Benutzern – Events, Blog, persönliche Meinungen
- **Prozess, Methoden, Tools:** Informationen und Ressourcen vermitteln. Zum Beispiel als Hilfestellung für das eigene Projekt. Prozess aus der Bachelorarbeit von Nico Iseli darstellen. Erstellung des *Empfehlungs-System*

5.1.1.2 Empfehlungs-System

Das Empfehlungs-System ist ein zentraler Punkt der Web-Applikation und es soll Benutzern helfen, geeignete Methoden für das eigene Projekt zu finden.

Das Empfehlungs-System besteht aus vier verschiedenen Elementen mit welchen den Benutzer interagieren kann um die am besten zu seinem Projekt passenden Methode zu finden. Um ein besseres Verständnis für die Funktionsweise des Empfehlungs-System zu erlangen, wurde eine erste Version auf eine Whiteboard aufgezeichnet und besprochen.

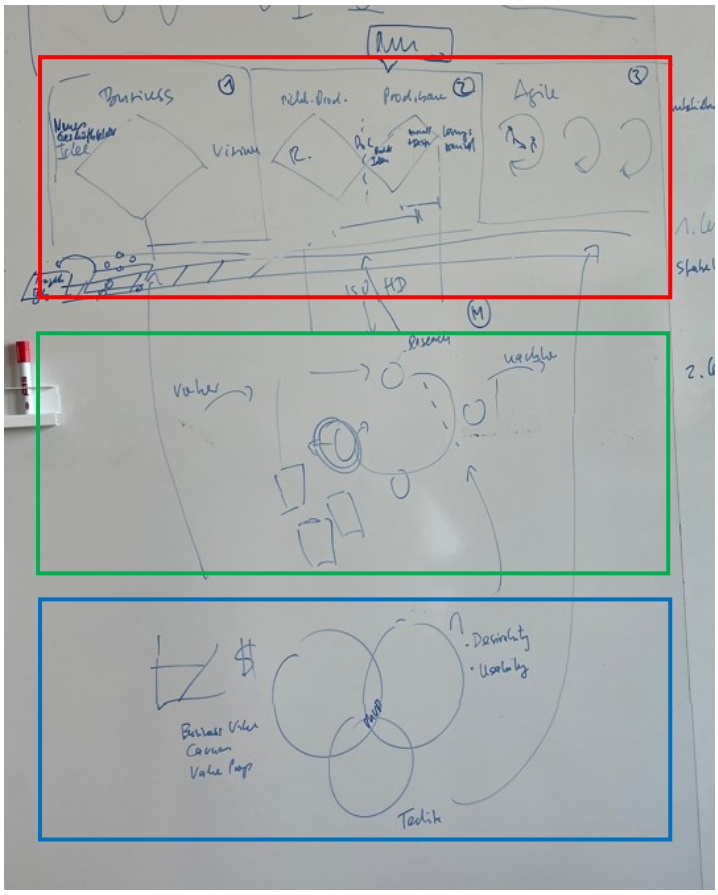


Abbildung 15: Empfehlungs-System erste Version

Gekennzeichnet in Rot ist ein Statusblock welcher anzeigt, in welchen Phasen das Projekt abgelaufen ist. Darunter soll ein Slider platziert werden, durch welchen noch genauer gezeigt werden soll, wo man sich in den jeweiligen Phasen befindet.
Beim grünen Block handelt es sich um die „Project-Engine". Hier können die einzelnen Methoden des Menschenzentrierten Gestaltungsprozesses eingesehen werden.
Der in blau gekennzeichnete Block hat die gleiche Funktionalität wie der rote Block. Hierbei handelt es sich aber um ein Venn-Diagramm.

Mehr Informationen zur Umsetzung der jeweiligen Blöcke sind im entsprechenden Kapitel unter *Empfehlungs-System* zu finden.

Als vierte Block wird der Tab „Empfehlungs-System" aus dem Figma-Prototypen aus Nico Iselis Bachelorarbeit genommen.

Abbildung 16: Figma - Wahl der Projektparameter

Hier können jeweils Projektparameter gewählt werden und die darunter angezeigten Methoden werden entsprechend gefiltert. Die in der Project-Engine angezeigten Methoden werden hierdurch auch beeinflusst. Auf die zwei anderen Blöcke wird nicht direkt Einfluss genommen.

5.1.1.3 Seiten

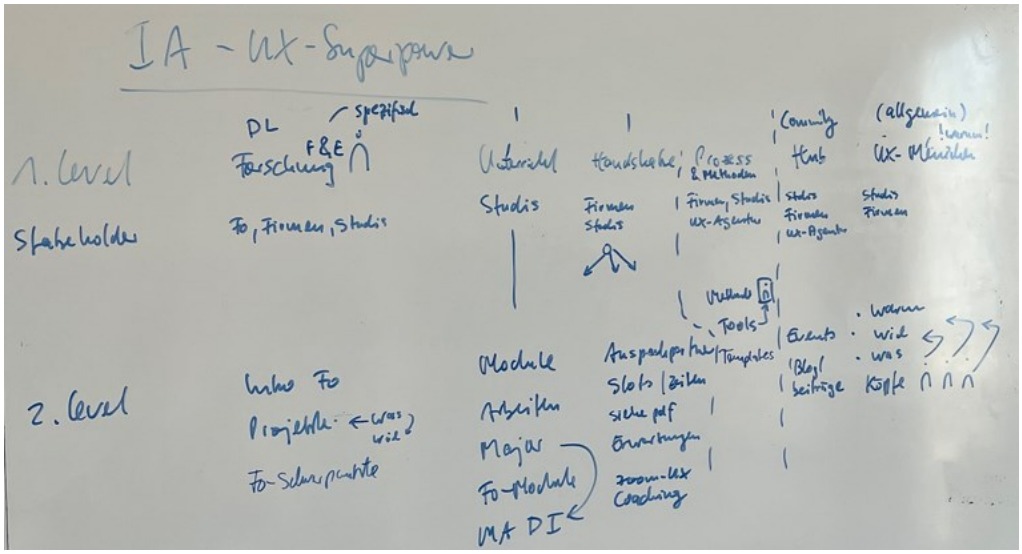


Abbildung 17: Geplante Navigation

Eine erste Version der Navigation der Webseite wurde anhand des Brainstormings und des daraus entstandenen Inhalts erstellt. Auf einem ersten Level der Navigation wurden die folgenden Themen definiert. Die jeweilige Umsetzung dieser Themen ist unter dem Kapitel *Web-Applikation* gefunden werden:

- *Forschung*
- *Unterricht*
- *Handshake*
- *Prozess & Methoden*
- *Community*
- *Allgemein*

5.1.1.4 Voraussetzungen an das CMS

Dank dem Workshop konnten nun auch die Voraussetzungen der UX Research Group an die gewählte CMS-Lösung definiert werden. Die wichtigsten Voraussetzungen sind folgend aufgelistet:

- **Einfaches Editieren von Dokumenten**: Das Bearbeiten von Dokumenten (Seiten, Inhalten) im CMS muss intuitiv und für die Mitarbeitenden der UX Research Group gut verständlich sein
- **Technisch machbar**: Die Umsetzung der gewünschten Features muss technisch machbar sein
    - Prototyp aus der Bachelorarbeit von Nico Iseli muss machbar sein
    - Funktionen des Empfehlungs-Systems
- **Gut erweiterbar:** Auf der gewählten Technologie soll in Zukunft in weiteren Projekten aufgebaut werden. Die Instandhaltung soll nur minimalen Aufwand und nicht tiefes, technisches Knowhow bedingen.
- **Wird aktiv unterstützt:** Das gewählte Produkt soll ausreichenden Ressourcen für Unterstützung anbieten. Zum Beispiel über eine Dokumentation, ein Community Forum oder YouTube Videos. Das Produkt soll zu dem auch noch aktiv von dessen Erstellern unterstützt und verwaltet werden. Ein

Produkt, welches sich bereits an seinem End-Of-Life befindet, eignet sich nicht.

5.2 CMS-Auswahl

In einem nächsten Schritt wurden CMS-Lösungen anhand den *oben* genannten Voraussetzungen *evaluiert und drei davon wurden den Mitarbeitenden der UX Research Group vorgeschlagen und zum Testen zur Verfügung gestellt.*

5.2.1 Erste Runde

In einer ersten Runde wurden drei verschiedene CMS-Arten evaluiert. Diese werden in den folgenden Kapiteln näher erläutert. Den Mitgliedern der Forschungsgruppe wurde ein Dokument gesendet in welchem die Gedanken des Autors, einige allgemeine Informationen sowie die Logindaten zum CMS dokumentiert wurden. Eine zensierte Version (ohne Logindaten) dieses Dokuments ist im Anhang unter *Erste Runde* zu finden.

5.2.1.1 Webflow[20]

Die erste CMS-Lösung, welche evaluierte wurde war Webflow. Webflow ist an sich kein traditionelles CMS, hat jedoch auch die Möglichkeit als solches zu agieren. Der Workflow mit Webflow ist ähnlich zu dem, wie der Workflow in Photoshop. Neue Seite können mit einem «What You See Is What You Get» (WYSIWYG) Editor erstellt werden. Die Core-Level Version von Webflow für 19 Dollar im Monat ermöglich die Erweiterung der Webseite mit eigenem Code[21].
Webflow wurde vor allem auf Grund der beruflichen Hintergründe und des Knowhows der UX Research Group vorgeschlagen.

5.2.1.2 WordPress[22]

WordPress ist die wohl bekanntestes CMS-Lösung und bietet deshalb eine grosse Community mit vielen Plugins und Themes (Designs). Die Erweiterung wird hierdurch vereinfacht. Viele verschiedene Plugins können jedoch auch Problemen beim Updaten sorgen, da hier oftmals auf Kompatibilitäten geachtet werden muss. Zudem sind diese Plugins und Themes auch nicht immer kostenlos was nicht geplante Kosten verursachen könnte.
WordPress ist eine Open-Source Software, das heisst der Source-Code kann beliebig bearbeitet werden und Designs können auch selbst erstellt werden. Das bringt auch den Vorteil das WordPress kostenlost verwendet werden kann und lediglich das Hosting einer Domain bezahlt werden muss.
Hier wurde auch das Plugin „Elementor" noch in Erwägung gezogen. Elementor stellt einen WYSIWYG-Editor zur Verfügung, was das Bauen von neuen Webseiten noch einmal angenehmer gestaltet. Das Plugin selbst kostet 49CHF pro Jahr. Zusammen mit geschätzten Hosting-Kosten von 15-20CHF im Monat würde sich die Kosten einer WordPress Lösung auf mindestens 15 CHF, oder entsprechend mit Elementor auf 64 CHF pro Jahr belaufen.

[20] https://webflow.com/
[21] https://webflow.com/pricing
[22] https://wordpress.org/

5.2.1.3 Prismic

Anders als WordPress ist Prismic ausschliesslich ein *Headless CMS*. Entsprechend musste hier auch noch eine Frontend-Technologie gewählt werden. Prismic besitzt keinen WYSIWYG-Editor für das Erstellen von neuen Seiten, sondern man hat als Entwickler die Möglichkeit sogenannte Slices zu erstellen. Mehr Informationen können im Kapitel *Prismic* gefunden werden

Das Entwickeln mit Prismic ist kostenlos und für bereits sieben ($7) Dollar im Monat kann eine Lizenz für bis zu drei Benutzern (CMS-Editoren) erworben werden.
Die Hosting-Kosten können hier je nach gewähltem Frontend noch variieren, mehr dazu ist im Kapitel *Prismic & NextJS* zu finden

5.2.2 Validierung und zweite Runde

Die vorher erwähnten CMS-Lösungen Workflow, WordPress und Prismic wurden von den Mitarbeitenden der UX Research Group angeschaut und individuel validiert. In einem zweiten Treffen mit der Forschungsgruppe wurden die entsprechenden Feedbacks der Mitarbeitenden aufgenommen und die nächsten Schritte definiert.
Webflow stellt sich nach persönlicher Erfahrung der Mitarbeitenden eher als komplizierte heraus. Kleine Änderungen am Design können eine ganze Seite durcheinanderbringen. Das Arbeiten mit Workflow wird nicht präferiert.

Laut Feedback scheint Prismic als Lösung interessant. Die Aufgesetzte Demo war jedoch nur auf Seiten des CMS vorhanden, also ohne ein entsprechendes Frontend in dem man die eigenen Änderungen sehen kann. Hier wurde entschieden, eine weitere Demo mit Prismic und einem entsprechenden Frontend zu erstellen. Die Wahl des Technologie Stacks sei hierbei dem Autoren überlassen. Es soll zudem auch noch eine neue Demo für ein *All-In-One CMS*, ähnlich WordPress, erstellt werden.
Diese beiden neuen Demos sollen jeweils ein Forschungsprojekt aufnehmen und im Frontend darstellen können.

5.2.2.1 OctoberCMS[23]

OctoberCMS basiert auf dem PHP Framework Laravel[24]. Die Software ist zwar Open Source und kann selbst gehostet werden, es muss jedoch eine Lizenz für 19 Dollar erworben werden. Eigene Templates können mit Twig[25], einer PHP Template engine, erstellt werden.

5.2.2.2 Prismic & NextJS

Allgemeine Informationen zu Prismic können dem Kapitel *Prismic* entnommen werden. Die Entscheidung zwischen NuxtJS und NextJS Framework fiel auf NextJS. Die Entscheidung basiert vor allem darauf, das NextJS auf React basiert. Das React Framework ist weiterverbreitet als Vue. Das heisst, für Entwickler stehen sehr viele Ressourcen zur Verfügung. Zudem wurde das React Framework von Meta (ehemalig Facebook) Open Source entwickelt und wird bereits seit 2013 von diesen verwaltet. Entsprechende wird ReactJS von einer Organisation verwaltete,

[23] https://octobercms.com/
[24] https://laravel.com/
[25] https://twig.symfony.com/

welche genügend Ressourcen hat um das Projekt zuverlässig und auch noch in Zukunft zu verwalten.
Es wurde das Starterkit Portfolio[26] aus der offiziellen Prismic Dokumentation eingesetzt. Diese Starterkit beinhaltet auch bereits einige Slices wie zum Beispiel für eine Hero-Section, ein Bild, oder einen Text und Bild. Diese Slices bieten uns bereits eine Grundlegende Funktion zum Bauen von Seiten und können auch später für die Umsetzung der Web-Applikation gut eingesetzt werden.

5.2.2.3 Statamic

Statamic wurde als zusätzliche Möglichkeit aufgenommen. Das CMS kostet 260 CHF im ersten Jahr, anschliessend jeweils 60 CHF pro Jahr. Das ist, vor allem im Vergleich mit den anderen CMS, sehr teuer. Die Developer Experience wurde hier jedoch als einiges angenehmer als die bei OctoberCMS empfunden, weshalb der Vorschlag als *All-In-One CMS* trotzdem auch noch gemacht wurde.

5.2.2.4 Auswahl

Den Mitgliedern der UX Research Group wurde erneut ein Dokument zugestellt, in welchem die Gedanken des Autoren, einige allgemeine Informationen sowie ein Login zum CMS dokumentiert wurden. Eine zensierte Version (ohne Logins) dieses Dokuments ist im Anhang unter *Zweite Runde* zu finden. Die verschiedenen Optionen wurden von den Mitarbeitenden individuelle evaluiert und in einem internen Meeting wurde entschieden, dass die Umsetzung der Web-Applikation mit Prismic und NextJS durchgeführt werden soll.

[26] https://prismic.io/docs/nextjs-example-projects

5.3 Hosting

Für das Hosting der Webseite empfiehlt sich Vercel[27]. Vercel ist die Firma, welche auch die gewählte Frontend-Technologie, NextJS, entwickelt.

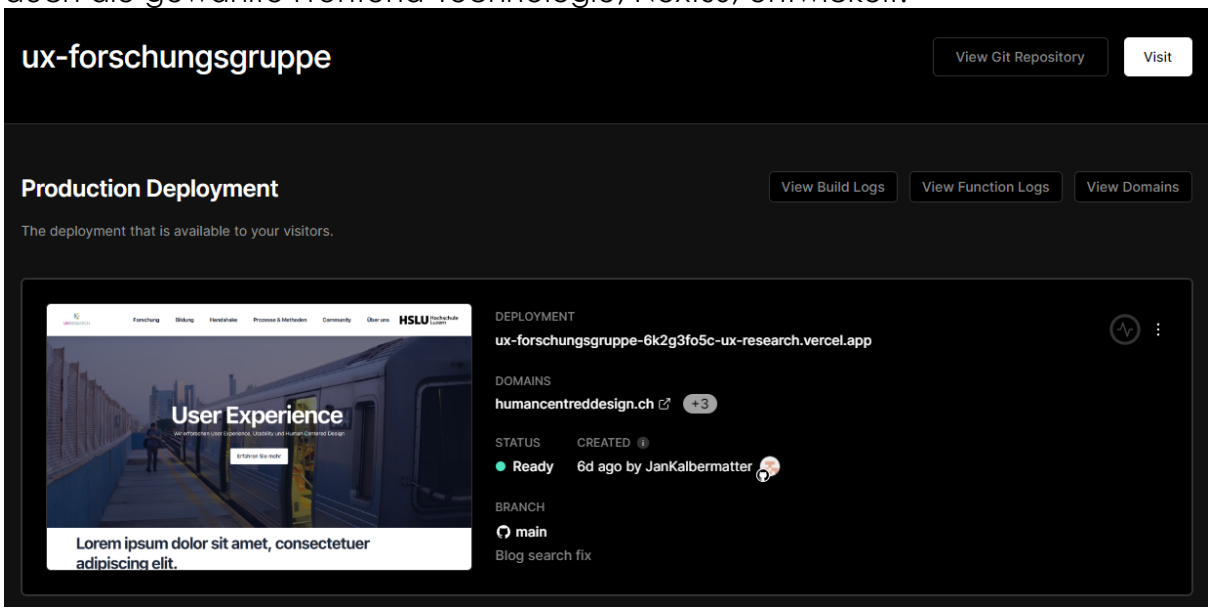


Abbildung 18: Web-Interface Vercel

Das Web-Interface zeigt einige Informationen zu der Webseite an wie Domain, der aktuelle Status und welche Commit auf welchem Branch in GitHub für den aktuellen Build verantwortlich ist. Wird ein Build gepushed, welcher nicht korrekt abläuft können hier Informationen dazu eingesehen werden. Zudem bleibt bei einem fehlerhaften Build die letzte Version der Webseite bestehen, so dass der Endnutzer trotzdem noch eine funktionierende Version der Webseite besuchen kann. Über Vercel gibt es zudem noch die Möglichkeit Environment-Variablen zu setzen, welche in NodeJS eingelesen und im Code der Web-Applikation eingesetzt werden können. Das ist praktisch für Features wie das Kontakt-Formular. Daten wie Passwörter können als Environment-Variable abgespeichert und dadurch vor dem Endbenutzer versteckt werden.
Vercel bietet die Möglichkeit, Webhooks zu erstellen. Bei einer Webhook handelt es sich um einen http-Callback welcher normalerweise durch irgendein Event, wie zum Beispiel dem Pushen von Code (automatischer Build erstellen) oder dem Ändern eines Dokuments im CMS ausgelöst wird. Wenn eines dieser Events vorkommt, wird von der Seite ein http-Request an die URL der Webhook gemacht. In unserem Fall werden diese Webhooks benutzt, um einen automatischen Build der Webseite auszulösen, falls auf das GitHub Repository gepushed wird. Prismic bietet auch die Möglichkeit Webhooks einzusetzen. Hier wird jeweils ein Build mit Vercel ausgelöst wenn ein Dokument im CMS geändert und veröffentlicht wird. (Wikipedia, 2022)

Vercel bietet einen grosszügigen „Hobby"-Plan an, welche sich für die Webseite gut eignen sollte. Der Plan stellt bis zu 100GB Bandbreite pro Monat zur Verfügung. Die meisten Daten werden Server-seitig geladen. Dadurch können API-Calls und damit auch die benutzte Bandbreite verringert werden. Falls die

[27] https://vercel.com/

Webseite sehr viel Traffic erhält kann bei Vercel auf den Pro-Plan geändert werden, welcher für 20 Dollar im Monat eine Bandbreite von 1TB im Monat bietet.

In Absprache mit Marcel Uhr wurden gemeinsam verschiedene Vorschläge gemacht und die Domain humancentreddesign.ch wurde auf Hostpoint im Namen von Marcel Uhr erworben und mit dem Hosting auf Vercel verbunden.

5.4 Web-Applikation

5.4.1 Systemarchitektur

Mit NextJS wurde einige Routen definiert, welche verschiedene Arten von Seiten voneinander trennen. Zum Beispiel müssen Forschungsprojekte anderes darstellt werden als Events, diese müssen wiederum anderes dargestellt werden als Blogbeiträge. Der Seitencontent wird mit ***getStaticProps*** aus dem CMS geladen. Mehr zu ***getStaticProps*** kann im *StaticProps* gefunden werden.

```
export async function getStaticProps({ locale, previewData }) {
  const client = createClient({ previewData });

  const page = await client.getByUID("page", "home", { lang: locale });
  const menu = await client.getSingle("menu", { lang: locale });
  const footer = await client.getSingle("footer", { lang: locale });
  const settings = await client.getSingle("settings", { lang: locale });

  return {
    props: {
      page,
      menu,
      footer,
      settings,
    },
  };
}
```

Abbildung 19: Index.js getStaticProps

Auf der Index.js Seite, welche sich auf dem Pfad / befindet, werden die oben gezeigten statischen Properties geholt. Die Konstante Client ist ein Objekt, welches die Verbindung zum CMS enthält. In dieser ***createClient***-Methode werden auch die einzelnen Pfade für die Webseite definiert.

```
export const createClient = (config = {}) => {
  const client = prismic.createClient(sm.apiEndpoint, {
    routes: [
      { type: "page", path: "/:uid" },
      { type: "methoden_picker", path: "/empfehlungs-system" },
      { type: "forschungsprojekt", path: '/projects/:uid'},
      { type: "methode", path: '/methods/:uid'},
      { type: "blog", path: '/blog/:uid'},
      { type: "autor", path: '/mitarbeiter/:uid'},
      { type: "event", path: '/events/:uid'},
      { type: "settings", path: "/" },
      { type: "navigation", path: "/" },
    ],
```

Abbildung 20: createClient Methode

Die in ***createClient*** definierte Routen sorgen dafür, dass die Seite zum Beispiel automatisch auf den Pfad ***methode/[Name der Methode]*** verlinkt, falls im Projekt auf ein Dokument des Typen

methode zugegriffen wird.

Anschliessend können über die verschiedenen Methoden des Prismic-Client-Objekts: ***getByUID***, ***getSingle***, ***getByType***, etc. die auf der Seite anzuzeigenden Daten geladen werden. Die Konstante ***page*** beinhaltet den eigentlichen Inhalt, welche auf der Seite angezeigt wird. *Menu* beinhaltet die Daten, welche zum Erstellen der Navigation gebraucht werden. Die ***footer*** Konstante beinhaltet die Daten zum Darstellen des Footers.
Jeglicher Code, welche als Template für eine Seite dienen soll, befindet sich im Ordner ***pages***. Zum besseren Verständnis für den Entwickler werden Dateien immer nach ihrem Pfad sortiert. Soll sich also eine statische Seite im Pfad ***/blog/MeineSeite*** befinden, wird diese sich im Ordner ***pages/blog/MeineSeite.js*** befinden.

5.4.2 Content Slices

Das Frontend wurde Komponenten-basiert aufgebaut, und durch die *Slicemachine* und *Prismic* können diese Komponenten auch den Autoren im CMS für den Einsatz zur Verfügung gestellt werden. Die Web-Applikation ist in Zukunft einfach erweiterbar, und es können beliebig nach Bedürfnissen neue Slices/Komponenten erstellen und in das CMS importiert werden.
Der Entwickler hat jeweils die Möglichkeit zu definieren, in welchen Dokument-typen (Seiten, Forschungsprojekte, Blogs, etc.) diese Slices auch eingesetzt werden können.

5.4.2.1 Vordefinierte Komponenten

Beim Aufsetzen von NextJS und Prismic wurde ein von Prismic erstelltes Starterkit eingesetzt. Diese Starterkit beinhaltet bereits einige Slices, welche eingesetzt werden können. Slices zum Erstellen von Text (ein oder zwei Spalten, mit oder ohne Bild), sowie eine Hero-Section, eine Quote und Bilder (mit Text und Link oder nur Bild) existieren bereist. Diese Komponenten können auch sehr gut für das Erstellen und Dokumentieren dokumentieren von Forschungsprojekten oder Blogs eingesetzt werden, weshalb hier durch dieses Starterkit einiges an Arbeit gespart werden konnte.

5.4.2.2 Eigene Komponenten

Die existieren Komponenten reichen für die Anforderungen der UX Research Group nicht aus.
Das Erstellen einer eigenen Komponente kann wie folgt zusammengefasst werden:

- Lokale Slicemachine Instanz starten.
- Über Slicemachine ein neues Slice erstellen und mit den nötigen Daten-Typen befüllen.
- Darstellung der Daten mittels NextJS umsetzen.
- Slice mit den Seiten verbinden, auf welche es benutzt werden kann.
- Lokale Slicemachine Instanz mit dem Prismic CMS synchronisieren.

Anschliessend kann das Slice zur Erstellung von Dokumenten im CMS eingesetzt werden. Eine genaue Erläuterung hierzu, ist im Anhang im *CMS-Handbuch* zu finden und an dieser Stelle wird nicht genauer darauf eingegangen.

5.4.3 Empfehlungs-System

Die einzelnen Blöcke, welche zusammen das Empfehlungs-System bilden und deren Umsetzung werden in den folgenden Kapiteln näher erläutert.

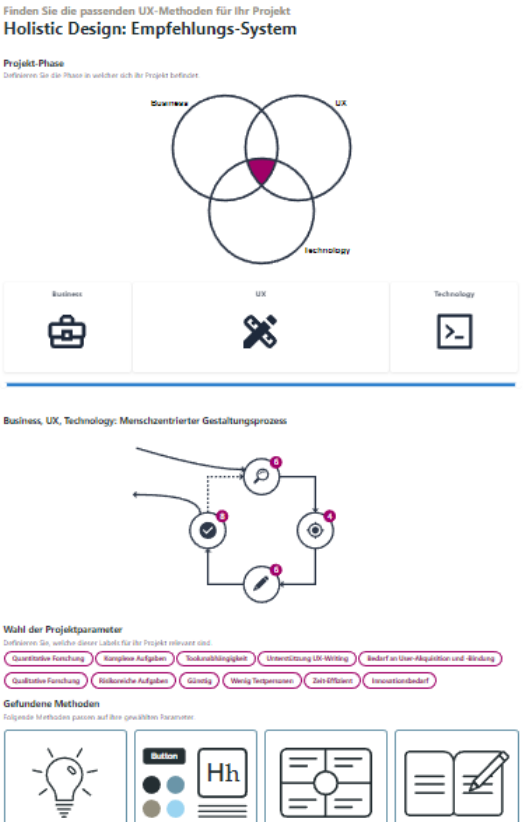


Abbildung 21: Empfehlungs-System

Die States (Zustände) der Blöcke im Empfehlungs-System, werden vom Empfehlungs-System selbst verwaltet und wird an die Blöcke nur zum Anzeigen weitergegeben. Zudem wurden für alle Elemente ***`onClick`***-Funktionen definiert, welche diese States manipulieren können. Somit können die Blöcke miteinander interagieren und den „globalen“ Empfehlungs-System State und dadurch die anderen Komponenten der Seite manipulieren.

5.4.3.1 StaticProps

Es wurde ein eigener Document-Type im CMS definiert, welcher für die Darstellung des Empfehlungs-Systems verantwortlich ist. Es handelt sich um einen Single-Document-Type, das heisst, es kann nur ein Dokument davon erstellt werden. Das Empfehlungs-System ist somit einmalig in der Web-Applikation vorhanden. (Abbildung 22, Zeile 200)

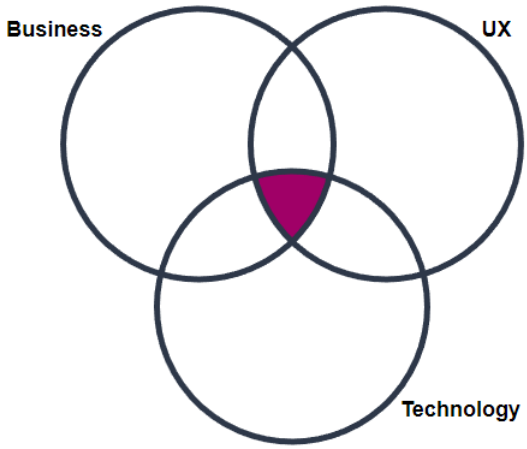
```
197  export async function getStaticProps({ previewData }) {
198    const client = createClient({ previewData });
199
200    const page = await client.getSingle("methoden_picker");
201    const menu = await client.getSingle("menu");
202    const footer = await client.getSingle("footer");
203
204    // Props for Components
205    const methodCriteria = await client.getAllByType('methodenkriterium')
206    // Add isActive = false to each criteria
207    methodCriteria.map(criteria => {
208      criteria.isActive = false;
209      return criteria
210    })
211
212    const methods = await client.getAllByType('methode')
213
214    return {
215      props: {
216        page,
217        menu,
218        footer,
219        methodCriteria,
220        methods
221      },
222    };
223  }
```

Abbildung 22: Empfehlungs-Systems Properties

Neben den Properties (Props) dieser Single Empfehlungs-System Typen, wurden auch noch, wie auf jeder Seite, die Daten des Menus und des Footers geholt (Abbildung 22, Zeile 201-202).
Alle Methoden, welche im CMS aufgenommen wurden, müssen im Empfehlungs-System auffindbar sein. Deshalb müssen diese auch als ***StaticProps*** geladen werden (Abbildung 22, Zeile 213). Zusätzlich werden noch die im CMS aufgenommenen Methoden-Kriterien geladen und es wird jeweils ein Boolean dem Methoden-Kriterium-Objekt hinzugefügt, welcher angibt, ob das aktuelle Methoden-Kriterium als aktiv oder als inaktiv angezeigt werden soll.

5.4.3.2 Venn-Diagramm

Mit dem Venn-Diagramm hat der Benutzer die Möglichkeit, die aktuelle Phase seines Projekts einfach auszuwählen. Ein erster Versuch, das Venn-Diagramm umzusetzen war es, die SVG der Grafik selbst zu erstellen (wie auch für die *Engine*). Vor allem aber die Überlappungen der SVG hat hier für viel Komplexität gesorgt, weshalb sich diese Angehensweise nicht für die Umsetzung eignet.



Die schlussendliche Umsetzung des Diagramms erfolgte mit dem Plugin UpSet.js[28]. UpSet ist eine JavaScript Library, welche es ermöglich Datensets mit mehr als 3 verschiedenen Datensätzen interaktiv zu visualisieren. Unter anderem auch mithilfe eines Venn-Diagramms. UpSet.JS hat zudem auch einen Adapter für die ReactJS Library, weshalb die Kompatibilität mit NextJS auch sichergestellt ist.

Abbildung 23: Prozess Venn-Diagramm

```
15      const [hasMounted, setHasMounted] = useState(false);
16      useEffect(() => {
17          // This will only be called once the component is mounted inside the browser
18          setHasMounted(true);
19        }, []);
20
21      if (!hasMounted) {
22          return null;
23      }
24
25      return (
26          <div className="ProcessVenn flex items-center justify-center w-full md:h-96 sm:h-96 h-80 ">
27              <VennDiagram
28                  tooltips={false}
29                  exportButtons={false}
30                  sets={sets}
31                  combinations={combinations}
32                  width={780}
33                  height={400}
34                  selection={selection}
35                  selectionColor={'#a00067'}
36                  onClick={(newSelection) => {
37                      onClick(newSelection)
38                  }}
39                  />
40          </div>
41      )
42  }
```

Abbildung 24: Hydration-Error Workaround

Das Einsetzen diese Library erfolgte jedoch nicht ganz ohne Probleme. Das Verwenden des Venn-Diagramms sorgte zu Beginn immer für einen *Hydration*-Error. Ein Workaround zu diesem Problem konnte auf StackOverflow gefunden werden[29]. Wie in der Abbildung 24 zu sehen ist, wird anstelle der Komponente nur der Wert ***null*** (Abbildung, Zeile 21-23) zurückgegeben. Erst so bald ***useEffect*** (Abbildung, Zeile 16-19) aufgerufen wird, kann die eigentliche Komponente, als das Venn-Diagramm, angezeigt werden, da ***useEffect*** erst dann ausgelöst wird, wenn die Komponenten bereits gerendert wurde. Durch das Warten auf diesen Render kann ein Hydration-Error verhindert werden, und das Plugin kann zur Umsetzung der Komponente eingesetzt werden.

[28] https://upset.js.org/
[29] https://stackoverflow.com/questions/74022328/how-to-solve-react-hydration-error-in-next-js-when-using-uselocalstorage-and

5.4.3.3 Status

Abbildung 25: Prozess Status-Block

Der Status Block zeigt wie auch das Venn-Diagramm, die aktuelle Phase des Projekts an. Nicht aktive Phasen, also Phasen dessen Methoden herausgefiltert werden, sind in diesem Block ausgegraut (Siehe Abbildung, Technology). Der Slider unter dem Block erlaubt es dem Benutzer, die aktiven Phasen anzupassen.

Die Umsetzung der einzelnen Blöcke (Business, UX und Technology) über dem Slide war sehr einfach. Der Slider stellte sich jedoch als schwieriger heraus. Ein Range-Slider mit zwei Angriffspunkte (links und rechts) existiert standardmässig nicht und muss deshalb entweder selbst erstellt werden oder es musste ein Plugin eingesetzt werden. Die eigene Umsetzung erwies sich hier als schwierig, jedoch nicht zu schwierig. Es wurden zwei verschiedene Slider platziert und diese wurden mittels CSS übereinander dargestellt. Anschliessend mussten die Slider-Werte evaluiert werden, und die beiden Angriffspunkte des Slider entsprechend angepasst werden.
Später im Projekt wurde die Chakra UI[30] Library hinzugefügt. Dabei handelt es sich um eine Komponenten-Library für ReactJS. Es werden einige Komponenten, wie Nachrichten-Boxen, Inputs und auch Slider zur Verfügung gestellt. Chakra UI biete auch einen Range-Slider an, welcher wesentlich unkomplizierter implementiert werden konnte. (siehe Abbildung 26)

```
33      <RangeSlider aria-label={ariaLabel} defaultValue={defaultValue} onChangeEnd={onSliderChangeEnd} isDisabled={!editable}>
34          <RangeSliderTrack>
35              <RangeSliderFilledTrack />
36          </RangeSliderTrack>
37          <RangeSliderThumb index={0} />
38          <RangeSliderThumb index={1} />
39      </RangeSlider>
```

Abbildung 26: Implementation Range-Slider von ChakraUI

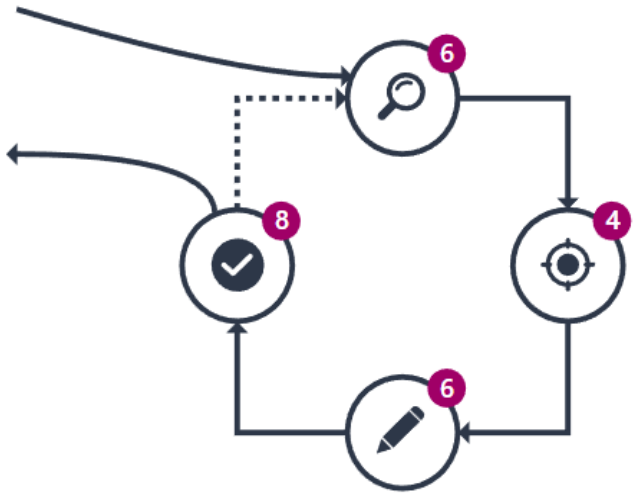
5.4.3.4 Engine

Die im Rahmen des Projekts sogenannte Prozess-Engine, kann dafür eingesetzt werden, die Methoden aufgeteilt auf die vier Prozess Schritte Forschung, Definition, Design und Validation einzusehen. Sie dient als eine der zwei Möglichkeiten, sich die Resultate der eigenen Filter im Empfehlungs-System anzusehen. Die Engine wurde mittels Scalable Vector Graphics (SVGs) umgesetzt. Es handelt sich um sehr einfache Formen (Kreise, Linien), welche durch wenig Aufwand mit HTML <svg>-Tags erstellt werden konnten. Die Logos, welche in den

Abbildung 27: Prozess-Engine

[30] https://chakra-ui.com/

Kreisen platziert werden, konnten aus dem Prototypen der früheren Bachelorarbeit herausgenommen, und anschliessend im Code korrekt platziert werden.



Die einzelnen Kreise der Engine sind jeweils mit Zahlen oder gar nicht gekennzeichnet. Eine angezeigte Zahl gibt an, wie viele Methoden durch einen Klick gefunden werden können.
Das ist besonders praktisch für den Einsatz im Empfehlungs-System, um dadurch dem Benutzer anzuzeigen das einer seiner gewählten Filter die vorhandenen Methoden beeinflusst hat.

Wird auf dem SVG auf einen der Kreise geklickt, so öffnet sich ein Modal, welches die jeweiligen Methoden des ausgewählten Teilschrittes anzeigt. Die angezeigten Methoden verlinken jeweils auf die Übersichtsseite der Methden. Das Modal kann entweder durch den „Schliessen"-Button oder durch einen Mausklick ausserhalb des Modals geschlossen werden.

Abbildung 28: Methoden-Modal

Dieser Block wird von allen Filtern im Empfehlungs-System beeinflusst und zeigt fast dieselben Methoden an, welche auch zuunterst im Empfehlungs-System angezeigt werden. Der einzige Unterschied hierzu ist das Anzeigen der Methoden in einem Modal und die Aufteilung in die vier Teilschritte Forschung, Definition, Design und Validation. Methoden der beiden Teilschritte Planung und Finalisierung werden nicht angezeigt und können nur zuunterst auf der Seite gefunden werden.

5.4.3.5 Projektparameter

Abbildung 29: Projektparameter

Die Projektparameter sind die letzte Art von Filter, welche dem Benutzer zur

Verfügung stehen. Diese Projektparameter werden im CMS definiert und die Komponente erstellt für jeden existierenden Paramater einen Button. Diese Buttons können von Benutzer ein- und ausgeschalten werden, um die jeweiligen Filter zu aktivieren und deaktivieren.
Die Umsetzung dieser Komponente war einfach. Die Komponente erhält ein Array aller Kriterien-Objekte. Diese Objekte besitzen jeweils ein Property ***isActive.*** Ist dieses Property ***true,*** so wird der Button Magenta eingefärbt. Ist es ***false*** bleibt der Button Weiss. Die Komponente dient nur dazu, die Buttons anzuzeigen. Eine ***onClick*** Methode, welche die Properties der Kriterien anpasst sowie die Kriterien selbst werden alle von einer Parent-Komponente zur Verfügung gestellt (in diesem Fall dem Empfehlungs-System).

5.4.3.6 Gefundene Methoden

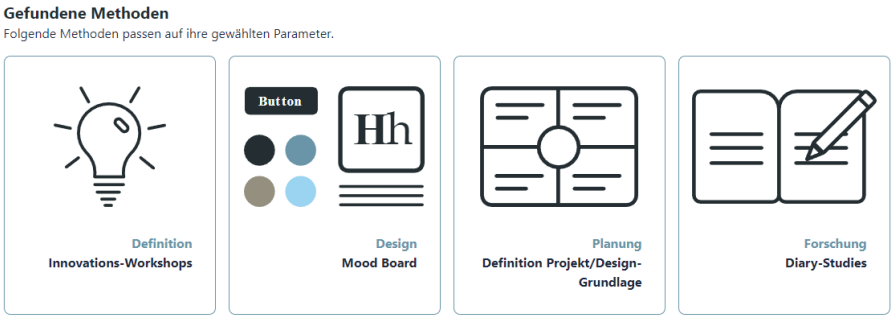


Abbildung 30: Methoden-Cards

Zuunterst im Empfehlungs-System werden die anhand der Filter gefundenen Methoden angezeigt. Diese werden jeweils bei einem Mouse-Hover mit einem Schatten ergänzt, um dem Benutzer anzuzeigen, dass die einzelnen Karten anklickbar sind. Die Komponente erhält die anzuzeigenden Methoden vom Empfehlungs-System als Property mit.
Das Design der Methoden in der Mobile-Ansicht wurde angepasst, um diese kompakter darzustellen und dem Endbenutzer ewiges Scrollen zu ersparen.

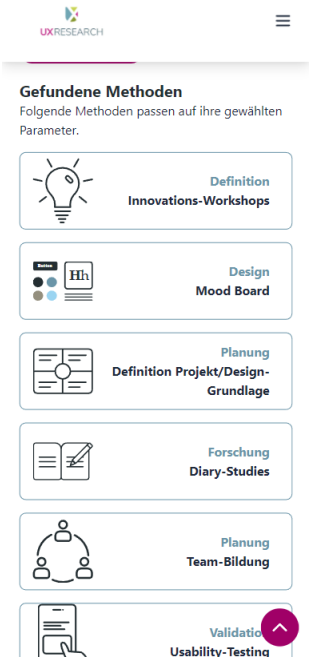


Abbildung 31: Methoden-Card Mobile Ansicht

5.4.3.6.1 Methoden anschauen

Wird eine Methode ausgewählt, öffnet sich eine Seite, welche weitere Informationen zu der gewählten Methode anzeigt. Die Methoden-Seiten, welche bereits im Figma-Prototypen existieren, wurden bereits alle im CMS aufgenommen und werden im Frontend entsprechend angezeigt. Beim Aufnehmen einer neuen Methode im Backend werden immer gewisse Standard-Daten aufgenommen.

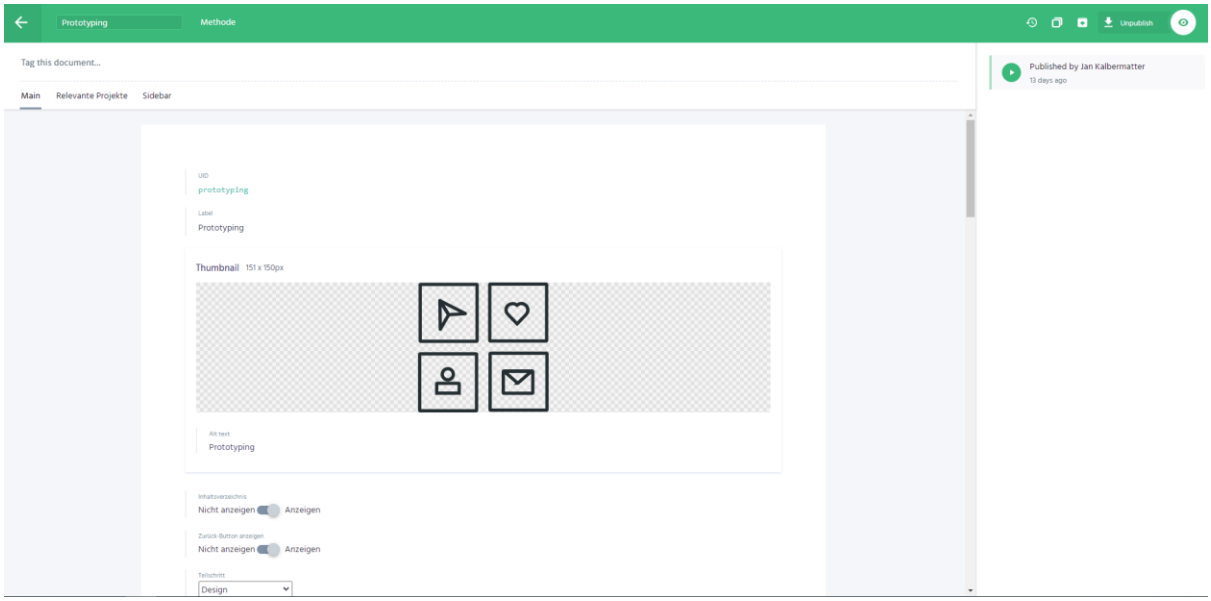


Abbildung 32: Bearbeiten der Methode "Prototyping" im CMS

Die Standard-Daten beinhalten ein Label, ein Thumbnail und zwei Toggle-Buttons um das Inhaltsverzeichnis respektive den Zurück-Button auf der Seite ein- und auszublenden. Zudem wird definiert, in welchem Teilschritt sich eine Methode befindet, zu welcher Projekt-Phasen eine Methode gehören kann und welche Methoden-Kriterien von der Methode erfüllt werden.

Einer Methoden können zusätzlich Slices hinzugefügt werden, welche die Editoren dazu einsetzen können, die Methoden-Ansicht mit Text, Bild und Video möglichst verständlich darzustellen.

In den beiden weiteren Tabs (relevante Projekte, Sidebar) können weitere Informationen zu den Projekten aufgenommen werden:

- **Relevante Projekte**: Forschungsprojekte mit der Methode verbinden, welche diese Methode verwenden. Diese Projekte werden in der Methoden-Ansicht zuunterst angezeigt.
- **Sidebar**: Es können im CMS Sidebars aufgenommen werden, welche dann beliebig auf Seiten eingesetzt werden können. Methoden können, wie Seiten auch, eine Sidebar anzeigen.

5.4.4 Kompetenzen vermitteln

Ein wichtiger Punkt aus dem ersten Workshop (siehe *Workshop – Vision der UX Research Group*)war das Bedürfnis der UX Research Group, ihre eigenen Kompetenzen mithilfe der Web-Applikation zu vermitteln und auch Werbung in eigener Sache (potenzielle Zusammenarbeit mit der Forschungsgruppe, UX-Module und Angebote der HSLU) zu machen.
Die eigentlichen Seiten zur Kompetenzvermittlung werden noch von den Mitarbeitenden der UX Research Group erstellt, ein Grundgerüst dafür wurde jedoch gelegt. Mitarbeitende können im CMS aufgenommen werden. Gleich wie bei den Methoden kann hier ergänzend zu den Standard-Daten auch noch Inhalt erstellt werden. Dieser Inhalt könnte dafür eingesetzt werden, die Mitarbeitenden

dem Benutzer näher zu bringen und deren spezifischen Kompetenzen zu vermitteln. Zudem können hier auch Forschungsprojekte angezeigt werden, an welchen der entsprechende Mitarbeitenden mitgewirkt hat.

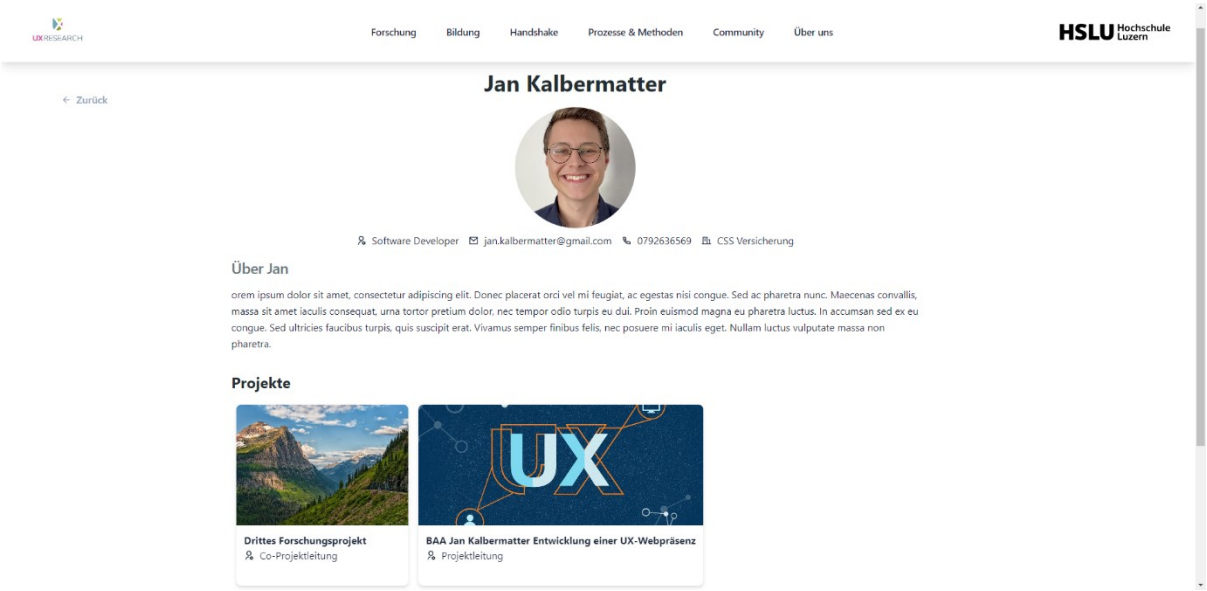


Abbildung 33: Mitarbeitenden-Ansicht

5.4.5 Forschungsprojekte

Forschungsprojekte, welche durchgeführt wurden, können auch im CMS aufgenommen und im Frontend dargestellt werden.

5.4.5.1 Darstellung

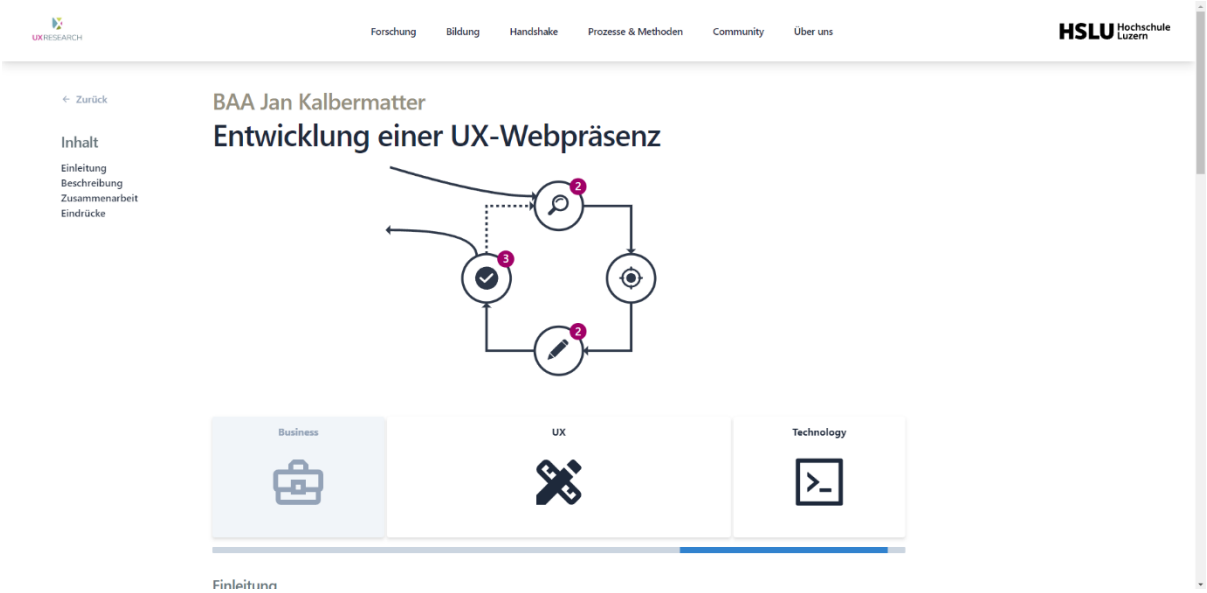


Abbildung 34: Forschungsprojekt

Anhand der Standard-Daten eines Forschungsprojekts wird eine Projekt-Engine (Siehe *Engine*) und eine Projekt-Status (siehe *Status*) Komponente hinzugefügt. In der Projekt-Engine werden die für das Projekt verwendeten Methoden angezeigt. Eine Methode kann hier auch mehrmals vorkommen, zum Beispiel einmal in der Forschung und ein weiteres Mal bei der Validierung. Mit dem Range-Slider der

Projekt-Stauts Komponente kann nicht interagiert werden. Anders als im Empfehlungs-System ist die Aufgabe der Projekt-Status Komponente nur das Anzeigen des Status und nicht auch noch für das Filtern von Methoden. Links zu sehen auf der Abbildung ist ein Inhaltsverzeichnis und ein Zurück-Knopf.

```
const TableOfContents = ({ title = "Inhalt", slices, additionalAnchors = null, className }) => {
    const slicesWithHeaders = slices.filter((slice) => (
        slicesWithAnchors.includes(slice.slice_type)
    )) // Slices welche nicht Anchors enthalten können herausfiltern

    const buildAnchor = (text) => text.replace(/\W+/g, '-').toLowerCase() // keine Abstände und lowercase

    let anchors = slicesWithHeaders.map((slice) => {
        let generatedAnchors = []
        if(!(slice.primary.text) && !(slice.primary.title)) return;

        generatedAnchors.push(slice.primary.text?.map(elem => { // Text analysieren
            if (elem.type !== 'heading3') return
            return {key: buildAnchor(elem.text), text:elem.text}; // neuer Anchor für h3
        }).filter(e => !!e))

        generatedAnchors.push(slice.primary.title?.map(elem => { // Titel analysieren
            if (elem.type !== 'heading3') return
            return {key: buildAnchor(elem.text), text:elem.text}; // neuer Anchor für h3
        }).filter(e => !!e))

        return generatedAnchors
    }).flat().filter(e => !!e).flat() // Lehre Elemente (undefined) entfernen und Array flatten

    if(additionalAnchors) { // zusätzliche Anchors als Property
        anchors = [...anchors, additionalAnchors]
    }

    const tocLinks = anchors.map((anchor, index) => {
        return (
          <li
            className={"font-medium"}
            key={`toc-link-${index}-${anchor?.key}`}
          >
            <Link href={`#${anchor?.key}`} className=""  replace passHref>
                <a className='hover:text-blues-400 anchor'>{anchor?.text}</a>
            </Link>
          </li>
        )
      }
    )
```

Abbildung 35: Inhaltsverzeichnis

Das Inhaltsverzeichnis kann auch bei Methoden und Seiten verwendet werden. Der Text-Inhalt, welcher im CMS mit Text oder Text-mit-Bild Slice aufgenommen wurde, wird beim Erstellen der Seite eingelesen, alle <h3>-Tags werden mit einer ID ergänzt und es wird eine Verlinkung im Inhaltsverzeichnis erstellt.

5.4.5.2 Forschungsprojekte finden

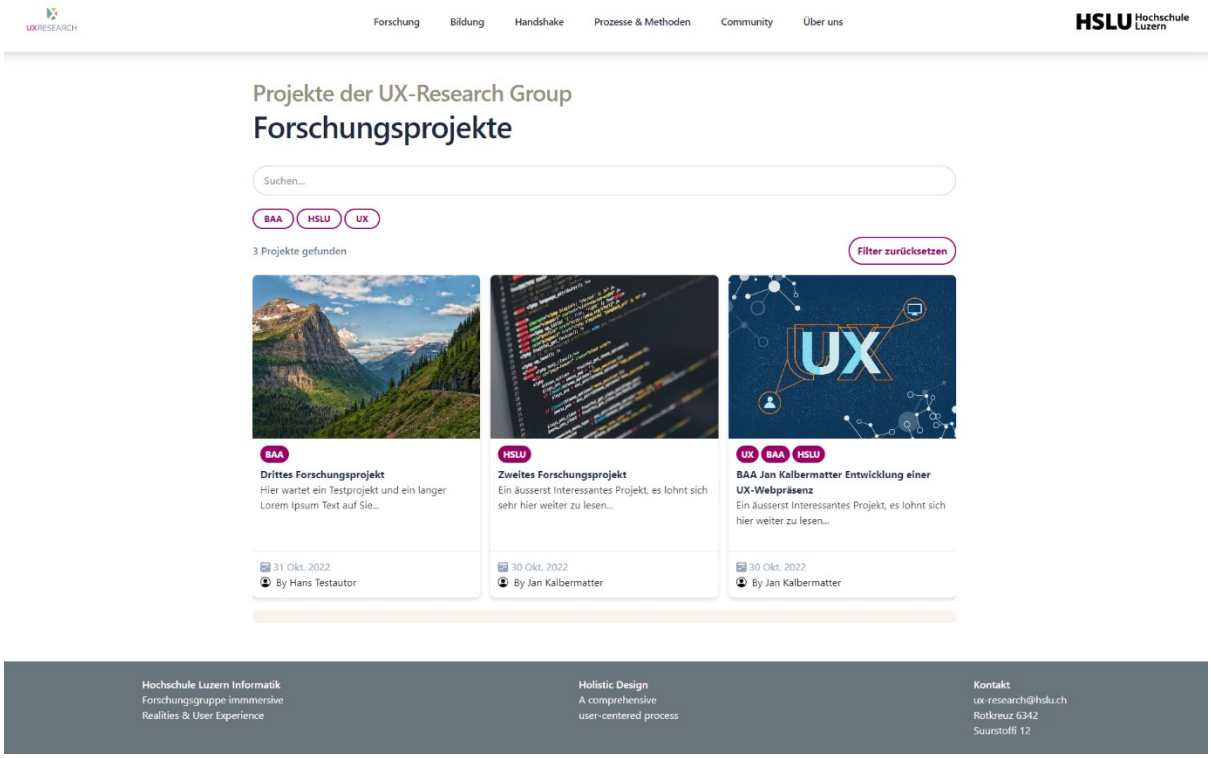


Abbildung 36: Forschungsprojekte

Die im CMS aufgenommen Forschungsprojekte werden wie in Abbildung 36 zu sehen aufgelistet. Der Benutzer hat die Möglichkeit über ein Such-Input oder über Tags Projekte zu filtern. Bei grossen Mengen an Projekten wird eine Scroll-bar entlang der X-Achse angezeigt. Forschungsprojekte können von links nach rechts durchgescrollt werden.

In einem ersten Anlauf wurde hier ein Pagination erstellt und im CMS konnte angegeben werden, wie viele Projekte maximal auf einer Seite angezeigt werden sollen. Die Pagination an sich hat funktioniert, jedoch wurde mit den Filtern später noch weiter Komplexität hinzugefügt, welche zu weitere Bugs führte. Die Umsetzung der Pagination hat sich als recht aufwendig herausgestellt und der geschriebene Code war schwierig nachzuvollziehen.
Zudem wurden auch bei User-Interviews vorgeschlagen, dass sich hier eine Scroll-bar anbieten könnte. Aufgrund dieser beiden Punkte wurde die Pagination entfernt und es wurde eine Scroll-bar eingefügt. Die ganze Logik für die Projektübersicht beschränkt sich nur noch auf das Filtern der Projekte und die komplizierte Verwaltung von mehreren Seiten und den entsprechenden Projekten fällt weg.

```
30  const filterProjects = (projects, searchTerm) => {
31      searchTerm = searchTerm.toLowerCase()
32      let filteredProjects = projects.filter(project => {
33          return project.title.some(title => title.text?.toLowerCase().includes(searchTerm)) || project.teaser.toLowerCase().includes(searchTerm)
34      })
35
36
37      if(tags.every(tag => !tag.isActive)) {
38          return filteredProjects
39      }
40
41      return filteredProjects.filter(project => {
42          return project.tags.some((projectTag) => {
43              return tags.find(tag => {
44                  return tag.label === projectTag && tag.isActive
45              })
46          })
47      })
48  }
```

Abbildung 37: Forschungsprojekte filtern

Tippt der Benutzer etwas in den Such-Input oder klickt er auf einen Tag-Filter, so wird die Methode ***`filterProjekts`*** ausgeführt. Diese Methode kontrolliert zuerst, ob das Projekt existiert, welches den Suchbegriff entweder im Titel oder im Teaser-Text beinhaltet (Abbildung 37, Zeile 32-34). Falls kein Tag aktiv ist, werden die Projekte nur nach dem Suchbegriff gefiltert zurückgegeben und angezeigt (Abbildung 37, Zeile 37-39). Sind eine oder mehrere Tags ausgewählt, wird darauf geachtet, ob mindestens ein Tag des Projekts angewählt ist. Das Resultat dieser Aktion wird spätestens danach zurückgegeben und angezeigt (Abbildung 37, Zeile 40-47).

5.4.5.3 Neue Projekte erstellen

Wird im CMS ein neues Forschungsprojekt erstellt (siehe *CMS-Handbuch*), wird dem Benutzer die folgende Ansicht gezeigt und einige Standard-Daten werden abgefragt.

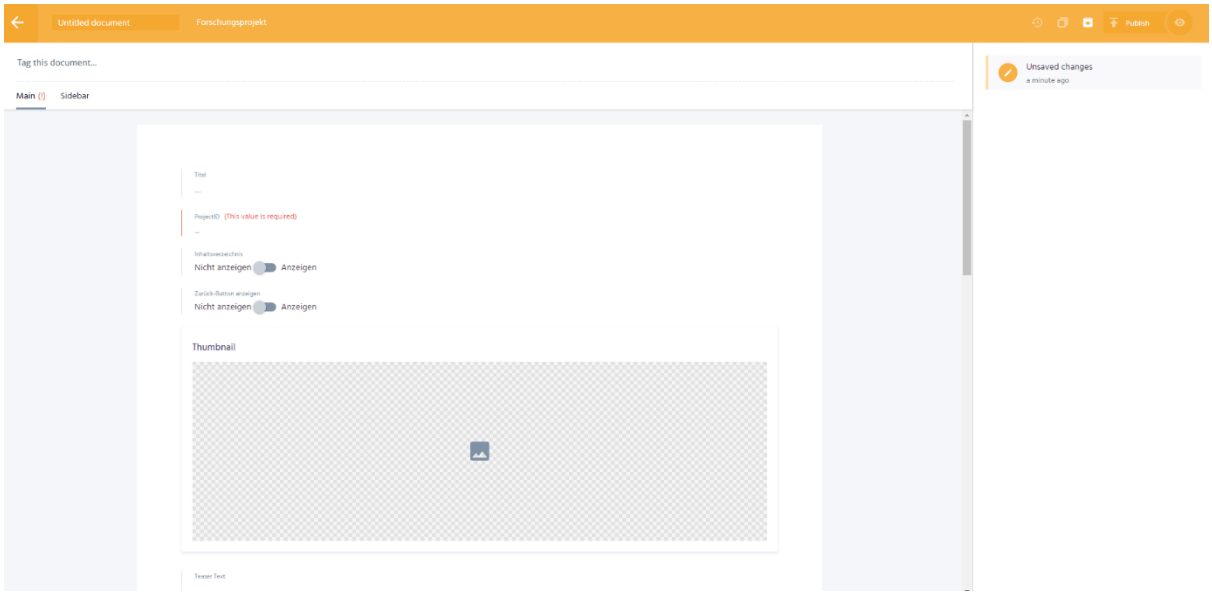


Abbildung 38: Prismic: Neues Forschungsprojekt

Es kann, wie bei Seiten und Methoden auch, ein Inhaltsverzeichnis und einen Zurück-Button aktiviert werden. Ein Thumbnail und ProjectID können auch gesetzt werden. Ein Teaser-Text, welcher in den Projekt-Cards (siehe Abbildung 26) angezeigt wird, hilft dem Benutzer einen kurzen Einblick in das Projekt zu erhalten. Für das Projekt kann zudem noch ein Start- und End-Datum definiert werden.
Im CMS aufgenommen Personen können mit einem Forschungsprojekt verbunden werden. Diese können entweder als Projektleiter, als Co-Projektleiter oder als

Projektmitarbeiter. Es können zudem noch keine bis viele Methoden mit einem Forschungsprojekt verbunden werden. Diese Methoden werden mithilfe der Projekt-Engine dem Benutzer angezeigt.
Neben den Standard-Daten können auch beliebig Slices hinzugefügt werden, um den Inhalt des Forschungsprojekts kompetent zu vermitteln.

5.4.6 Community

Community-Features wurde bereits im ersten Workshop (siehe *Workshop – Vision der UX Research Group*) sowie in der aufbauenden Bachelorarbeit von Nico Iseli als ein potenzielles Feature erwähnt.
Im Rahmen des Projektes wurde eine erste Version eines Event-Kalenders und eines Blogs erstellt. Potenzielle Erweiterungen dieser Features können im Kapitel *Ausblick* gefunden werden.

5.4.6.1 Events

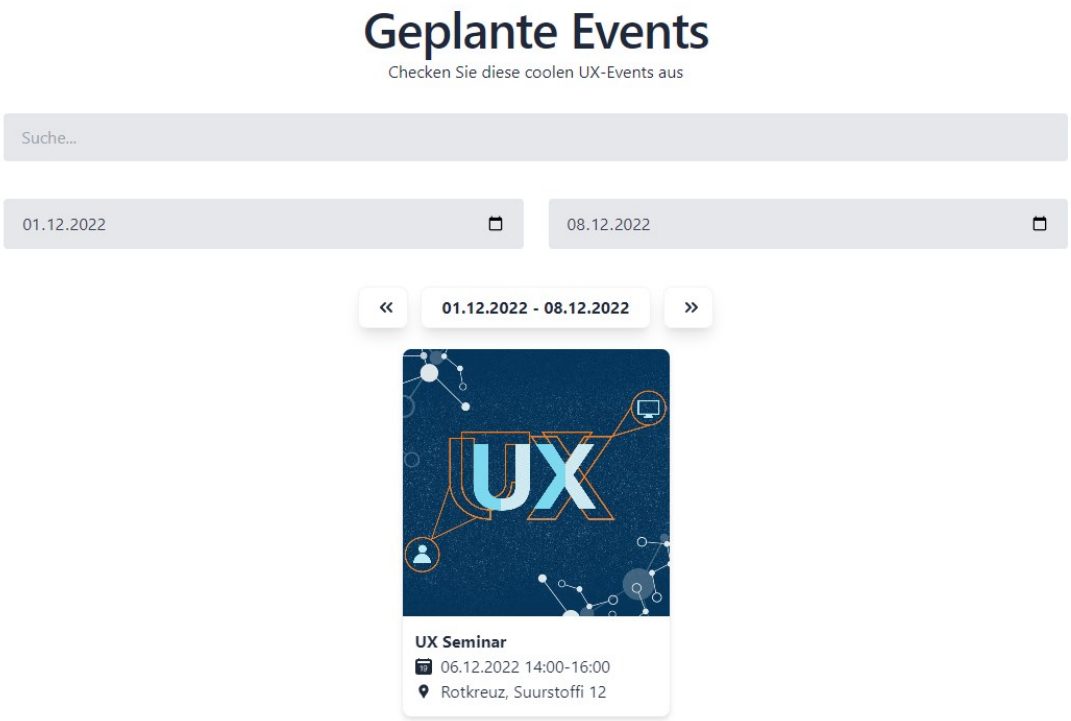


Abbildung 39: Events

Events werden wie auch Projekt und Methoden im „Karten-Format“ angezeigt. Hier werden einige Standard-Daten aus dem CMS angezeigt, um dem Benutzer einen ersten Überblick über das Event zu geben.
Mittels drei Inputs können die Events gefiltert werden. Bei den beiden Datumsfeldern handelt es sich um Inputs mit dem type=“date“ Property. Dadurch wird automatisch ein Datepicker erstellt, um die Auswahl des Datums zu vereinfachen. Die beiden Buttons mit den Pfeilen (<< und >>) navigieren jeweils Wochenweise die Events.
Wird ein Event angeklickt gelangt man zur Event-Ansicht, in welcher im das Thumbnail sowie mehr detaillierte Informationen zum Event angezeigt werden. Durch Slices kann der Inhalt dieser Seite auch erweitert werden und das Event kann den Kunden noch besser verkauft werden.

5.4.6.2 Blogs

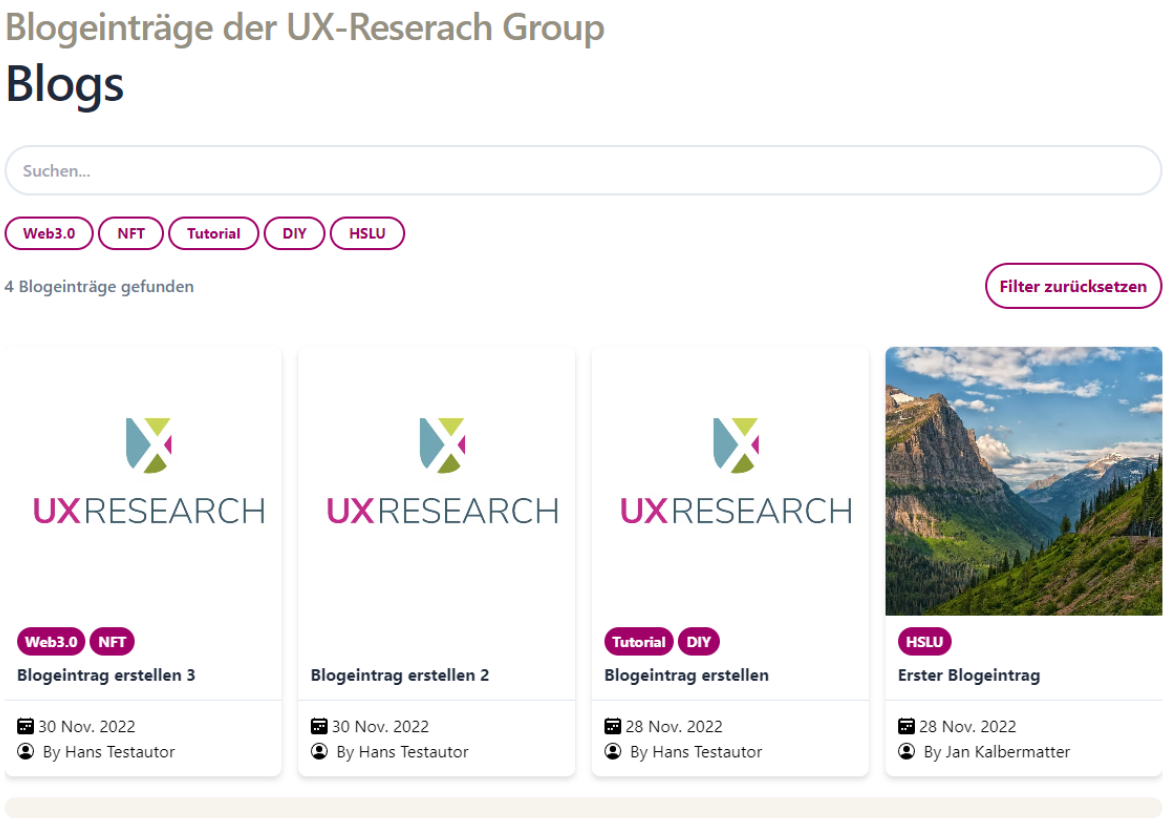


Abbildung 40: Blogs

Die Blog Übersicht ist an sich gleich gestaltet wie auch die die Forschungsprojekte (siehe *Forschungsprojekte finden*). Auch hier wurde im ersten Anlauf eine Pagination erstellt, basierend auf der Lösung aus den Forschungsprojekt Übersicht. Die Pagination wurde auch aus denselben Gründen verworfen und der selber Code aus der Forschungsprojekt-Übersicht konnte auch für die Forschungsprojekte verwendet werden.
Blogs können mittels eines Such-Inputs und den Tags gefiltert werden. Der Code für das Filtern der Blogs konnte basierend auf den Filtern der Forschungsprojekte erstellt werden (siehe Abbildung 27)

5.4.7 Zusammenarbeit

Die mögliche Zusammenarbeit von Benutzern mit der UX Research Group wurde bereits im ersten Workshop besprochen. Ein Kontakt-Formular wurde als Slice erstellt welches auf Seiten beliebig verwendet werden kann.

5.4.7.1 E-Mail versenden

Das Versenden von E-Mail ist der zentrale Punkt der Kontaktaufnahme und stellte sich als schwieriger als gedacht heraus.
Für die Umsetzung der E-Mail-Versand wurde das Node.JS[31] Modul NodeMailer[32] eingesetzt. Mit dem NodeMailer Modul kann ein sogenannter Transporter erstellt werden. Die Aufgabe es des Transporters ist das eigentliche versenden der E-Mail.

[31] https://nodejs.org/en/
[32] https://nodemailer.com/about/

Diese Transporter verbindet sich über SMTP bei Mailserver mit der angegebenen E-Mail-Adresse und dem Passwort (Abbildung, Zeile 5-16).

```
2   export default function (req, res) {
3       let nodemailer = require("nodemailer")
4
5       const transporter = nodemailer.createTransport({
6           port: process.env.MAIL_PORT,
7           host: process.env.MAIL_HOST,
8           auth: {
9               user: process.env.MAIL_FROM,
10              pass: process.env.MAIL_PASSWORD,
11          },
12          secure: false,
13          tls: {
14              ciphers: "SSLv3"
15          }
16      })
17
18      const mailData = {
19          from: process.env.MAIL_FROM,
20          to: process.env.MAIL_TO,
21          subject: `Nachricht von ${req.body.name} - ${req.body.email}`,
22          text: req.body.message,
23          html: `<div>${req.body.message}</div>`
24      }
25
26      transporter.sendMail(mailData, (err, info) => {
27          if(err) {
28              console.error(err)
29              res.status(500).send()
30          } else {
31              console.log(info)
32              res.status(200).send()
33          }
34      })
35
36  }
```

Abbildung 42: Nodemailer - E-Mails versenden

Die Option ***secure*** wird auf false gesetzt. Das sorgt dafür das NodeMailer für die Verbindung das TLS, anstelle des SSL-Protokolls eingesetzt. TLS kann eingesetzt werden, um eine Verbindung zu verifizieren und Daten verschlüsselt über diese zu senden. SSL kann als deprecated angesehen werden und TLS als den Standard. TLS ist eine neuer Version von SSL, welche einige Sicherheitslücken des alten Protokolls behebt.

Die durch NodeJS ermöglichten Enviroment Variablen (z.B. ***process.env.MAIL_POST***) können direkt über das Web-Interface von Vercel bearbeitet werden. Während der Umsetzung der Komponente wurde ein gratis E-Mail Account bei Gmail erstellt. Die E-Mail-Adresse **uxhslu.test@gmail.com** wurde erstellt und mit dieser Adresse werden alle E-Mails über das Kontakt-Formular

gesendet. Bei der Verwendung einer G-Mail Adresse ist auf einige Punkte zu achten.

Gmail legt besonderen Wert darauf, der der Benutzer welcher sich anmelden will eine echte Person ist. Dies könnte auch zu Bugs führen, falls sich der Standort des Server ändert, und Google annimmt das es sich um einen Highjacking-Versuch handeln muss.
Zudem hat Google Konzept „Less Secure“[33] eingeführt. Hat ein Benutzer die Less Secure Apps nicht aktiviert, so kann er sich nicht nur mit seinem Google-Account Password und Benutzername auf anderen Seiten anmelden.
Laut NodeMailer wäre es möglich diese Einstellung zu aktivieren, um dann den E-Mail-Account als „Bot-Account“ auf der Webseite einzusetzen. Seit der letzten Aktualisierung der NodeMailer Dokumentation hat jedoch Google „Less Secure“ entfernt. Das Login in Drittanbieter-Apps mit einem Google-Account ist nicht mehr möglich, weshalb das Versenden nicht nur damit umgesetzt werden konnte.
Statt dessen musste, wie in der NodeMailer Dokumentation beschrieben, eine 2-Faktor-Authentifizierung für den Google-Account aufgesetzt werden. Da diese Daten nach dem Abschluss des Projekts nicht beim Autor bleiben sollen, wurde die 2-Faktor-Authentifizierung mittels Backup-Codes aufgesetzt. Die Codes werden beim Login benötig und zusätzlich zum Benutzernamen und Password wird einer davon abgefragt. Bereits benutzte Codes sind nicht mehr gültig und können gelöscht werden. Wurden alle Backup-Code aufgebraucht, besteht auch die Möglichkeit neue generieren zu lassen.
Neben dem Aktivieren der 2-Faktor-Authentifizierung musste zudem noch auf Seiten von Google ein App-Passwort generiert werden. Anschliessend wird für die Anmeldung über NodeMailer nicht das eigentliche Passwort des Google-Accounts, sondern dieses generierte App-Passwort eingesetzt werden. (NodeMailer, 2022)

Nach anfänglichen Problemen bei der Umsetzung mit NodeMailer wurde noch in Absprache mit Marcel Uhr eine neue E-Mail-Adresse für die UX Research Group angefragt (**ux-research@hslu.ch**). Falls die Lösung mit dem Google-Account nicht funktioniert hätte, wäre es hier noch eine Möglichkeit gewesen, diese E-Mail-Adresse für das Versenden der E-Mail zu benutzen. Eine Weiterleitung der E-Mail an den HSLU-Account von Marcel Uhr wurde vom ServiceDesk eingerichtet, jedoch gab es Schwierigkeiten beim Hinzufügen der E-Mail in Outlook. Vor allem aus Zeitgründen wurde jedoch der Entscheid getroffen, für den Anfang den Google-Account einzusetzen, das diese Umsetzung zu diesem Zeitpunkt bereits feststand. Im Ausblick wird ein weiteres Mal auf die erstellte E-Mail-Adresse eingegangen.

5.4.8 Inhalt erstellen

Vor der Durchführung des *Feedback Workshop* wurden die Seite mit Inhalten befüllt, so dass den Teilnehmern auch eine vollständige Seite zum Evaluieren vor sich haben. Der grossteil des erstellten Inhalts waren die Methoden aus dem Figma-Prototypen. Diese Methoden sind wichtig, um die volle Funktionalität der

[33] https://support.google.com/accounts/answer/6010255?hl=en

Projekt-Engine und des Empfehlungs-Systems zu zeigen (Webseite - Empfehlungs-System).
Auch der restliche Inhalt aus dem Prototypen, wie zum Beispiel der Holistc Design Prozess (Webseite - Design Prozess) wurde im CMS übernommen und mit den vorhandenen Slices dargestellt.
Die Übersichtseiten für Projekte, Blogs und Events wurde auch noch einmal an Aufbau der restlichen Seite angepasst.
Für den Inhalt bezüglich der Module und des Bachelorstudiums wurde die offizielle HSLU zur Hilfe genommen. Nicht aller Inhalt konnte gleich wie auf der HSLU-Webseite aufgenommen werden. Hier könnte noch ein Bedürfnis auf mehr Slices aufkommen (siehe *Ausblick*). Die Seite mit den Modulbeschreibungen wurde besteht beispielsweise aus reinem Text, was für Benutzer recht unübersichtlich sein könnte.

5.4.9 Feedback Workshop

Die zuvor erarbeiteten Resultat wurde in drei weiteren Workshops mit Armin Egli und Rosina Brosi sowie mit drei potenziellen Benutzer angeschaut und evaluiert. Durch die verschiedenen Hintergründe der Workshop-Teilnehmer konnte zu verschiedensten Bereichen Feedback aufgenommen und umgesetzt werden.

Vor dem Workshop wurden einige Fragen notiert, die Workshops selbst liefen jedoch frei ab und die Fragen dienten nur dazu, einen roten Faden durch den Workshop zu haben. Zu Beginn des Workshops wurden den Teilnehmern ein kurzer Überblick über deren Funktionen gegeben.

Im *Ausblick* wird genauer auf potenzielle Erweiterungen eingegangen.

5.4.9.1 Workshop mit der Forschungsgruppe

Der erste Workshop wurde mit Armin Egli und Rosina Brosi der UX Research Group durchgeführt. Zusätzlich zum Frontend wurden den beiden Teilnehmer noch das CMS gezeigt.

Das Feedback der beiden Teilnehmer wurde zusammengefasst und die wichtigsten Punkte werden folgend aufgelistet:

Tabelle 6: Resultate - Workshop 2 mit Forschungsgruppe

Wunsch	Wurde umgesetzt	Begründung
Die Phasen-Blöcke ausgrauen, falls diese nicht aktiv sind		
Anpassung des Titels im Engin-Modal		
Möglichkeit ein Preview-Videos beim Hovern einer Methode anzuzeigen		Wurde nicht priorisiert.

Im Methoden-Modal direkt schon Tipps zu den Methoden anzeigen		Wurde nicht priorisiert.
Titel über der Projekt-Engine soll sich mit Venn-Diagramm anpassen		
Anzeigen «Keine Methode gefunden» falls Resultat im Empfehlungs-System leer		
Empfehlungs-System für Kunden einen Prozess generieren		Erweiterung des Empfehlungs-Systems. Wurde im Workshop angemerkt das es sich dabei eher für eine zukünftige Weiterentwicklung handeln soll. Siehe *Ausblick*.
Status und Engine in einem Projekt **immer** anzeigen (nicht nur als Slice hinzufügbar)		
Anzeigen von archivierten Blogeinträgen		Wird von Prismic nicht von direkt supportet. Siehe *Ausblick*
Breadcrumbs hinzufügbar		
Footer überall anzeigen		

Mit den Datentypen im CMS waren die Teilnehmer zufrieden. Es wurde jedoch Betont das eine Bedienungsanleitung erstellt wird, welche allen Mitgliedern die Einführung in Prismic vereinfachen und als Nachschlagewerk dienen soll.

5.4.9.2 Workshops mit Studierenden der HSLU

Der zweite Workshop wurde mit Aleksa Zivadinovic und Carina Vasconcales durchgeführt. Sie sind beide Informatik-Studierende an der HSLU im 3. Semester.

Das Feedback der beiden Teilnehmer wurde zusammengefasst und die wichtigsten Punkte werden folgend aufgelistet:

Tabelle 7: Resultate - Workshop 2 mit Studierenden

Wunsch	Wurde umgesetzt	Begründung

Anpassung der Navigation		Einige Titel waren unklar und entsprachen nicht den Erwartungen der Teilnehmer. Anpassungen wurden vorgenommen. Der Tab «Handshake» bleibt als solcher bestehen. Kann aber jederzeit über das CMS angepasst werden
Empfehlungs-System ist nicht sprechend genug		Der Name macht erst Sinn, sobald man die Seite sieht.
Klicks auf Projekt-Card ist nicht immer klar		Ein Hover-Effekt, welcher das Thumbnail an zoomt, wurde hinzugefügt.
Zurück-Buttons fehlen, sind komisch platziert.		Ein Zurück-Button kann für alle Seiten im Backend aktiviert werden. Erstellte Layouts sorgen für gleiches Alignment der Zurück-Bottons auf allen Seiten.
Methoden-Card ist in der Mobile-Ansicht unübersichtlich.		Ein separates Design für die Mobile Ansicht der Methoden-Cards wurde erstellt.
Inhaltsverzeichnis mit eingefärbtem Text		Links im Inhaltsverzeichnis werden in einem Blau-Ton aus dem Design eingefärbt.
Scroll auf der X-Achse anstelle einer Pagination. Teilnehmern fänden diese Lösung modernen/einfacher zu bedienen		Pagination wurde entfernt und durch einen Scroll-Bar entlang der X-Achse ersetzt.
Slider des Status-Block hat noch Griffe links und rechts, weshalb es so aussieht das dieser noch interaktiv ist. Den Teilnehmern war nicht klar, dass dieser Slider nicht funktioniert.		
Blogs wie Projekt anzeigen		Wurde entschieden nicht zu machen, Selbe Design könnte bei Benutzer für Verwechslungen sorgen
Datepicker mit Range für Events		Zeitlich nicht umgesetzt. Siehe *Ausblick*

Die Teilnehmer waren allgemein mit dem vorhanden und geplanten Inhalt der Seite zufrieden und hatten bezüglich Inhalt keine weitere Vorschläge. Das Design der Seite sähe auch ansprechen, wenn auch etwas simpel aus.

5.4.9.3 Workshop mit Berufstätiger Person im UX-Bereich

Der dritte und letzte Workshop wurde mit Annika Hovingh durchgeführt. Annika hat eine Ausbildung als Mediamatiker und arbeitet beim Kanton Luzern als UX-Designerin/Applikationsmanagerin. Aufgrund des Arbeits-Hintergrunds von Annika und dem bereist erhaltenen Feedback aus zwei Workshops wurde bei diesem Workshop wurde vor allem mehr Wert auf das UX der Webseite gelegt.

Durch den Workshop konnten einige Punkte geklärt werden, und das Allgemein Design der Webseite wurde noch aufgeräumt.

Folgendes Feedback wurde durch diesen Workshop aufgenommen

- Das Ausrichtung aller Element auf der Webseite, sollte sich auf einer Linie befinden.
    - Alle erstellten und noch zu erstellenden Slices
    - Inhalt in allen Cards (Methoden, Forschungsprojekt, etc.)
- Das Design des Empfehlungs-System ist zu breit und sollte nicht die ganze Webseite aufnehmen, damit es übersichtlicher ist.
- Zooms und Schatten für Hover-Effekte einsetzen. Dem Benutzer muss immer klar sein, wann er mit einem Element interagieren kann.
- Projekt eher mit einem Scroll als einer Pagination
    - Vielleicht sogar mit Pfeil-Navigation. Gewähltes Projekt ist im Zentrum und es kann durch ein Karousel navigiert werden
- Falls Filter vorhanden sind mit Elementen, welche darunter angezeigt werden, sollten diese Elemente maximal so breit sein, wie die Filter (Siehe z.B. *Forschungsprojekte finden*)

Zusätzlich zu diesem Feedback wurde noch erwähnt, dass die Seite simpel gehalten ist und es wird sich vom Teilnehmer mehr Farbe gewünscht. Der allgemeine Eindruck der Seite sei zwar der einer Schul-Webseite, da es sich aber um ein UX-orientierte Seite handelt, könnte man die Webseite noch lebhafter gestalteten. Als Beispiel wurde hierfür die im Projekt eingesetzt Akzentfarbe genannt (Magenta). Durch den verbreiteten Einsatz dieser Farbe und deren Farbtönen könnte man weitere Elemente noch genauer definieren, herausstechen und somit die Seite noch lebendiger wirken lassen.

6 Evaluation und Validation

Die Evaluation und Validation der im Projekt erzielten Resultat wird in den folgenden Kapiteln gehandelt.

6.1 Aufgabestellung

Die Aufgabenstellung und deren gesetzt Ziele werden folgend validiert:

#	Ziel (Kurzbeschrieb)	Status	Beweis
1	Bedürfnisse der Potentiellen Benutzer mit mindestes fünf Interview analysieren.	Erfüllt	Siehe *Feedback Workshop*
2	Bedürfnisse der Forschungsgruppe analysieren.	Erfüllt	Siehe *Workshop – Vision der UX Research Group*
3	Community Funktionen.	Erfüllt	Siehe *Community*
4	Die Webseite ist Responsive.	Erfüllt	Siehe *Responsive Design*
5	Prototyp wird mittels Blackbox und Usability Tests getestet.	Erfüllt	Siehe *Blackbox Tests*
6	Verschiedene Tools zur Erstellung der Webseite werden validiert	Erfüllt	Siehe *Usability Tests*
7	Hosting wird analysiert und durchgeführt	Erfüllt	Siehe *Hosting*

Tabelle 8: Validierung Aufgabenstellung

Die vollständige Aufgabenstellung kann im Anhang gefunden werden.

6.2 Blackbox Tests

Die abschliessenden Blackbox Test wurde von drei Nutzern und dem Autoren der Arbeit durchgeführt und die jeweiligen Testprotokolle können auch im Anhang gefunden unter Blackbox-Tests werden. Die Testprotokolle wurden alle vom Autor vereinheitlicht, damit diese auch dem Stil der Rest der Arbeit entsprechen. Aus den Testprotokollen ist zu erkennen, das die getesteten Funktionalitäten alle wie erwartete Funktionieren und auf der Webseite eingesetzt werden können.

6.3 Usability Tests

Wie die Blackbox Tests wurden die Usability Tests durch von drei Testpersonen, jeweils in Begleitung des Autors, durchgeführt. Die Protokolle der Usability-Test können im Anhang unter *Usability-Tests* gefunden werden.
Hier aufgeführt sind die wichtigsten Erkenntnisse aus den drei Testläufen

- **Hanshake:** Der Name „Handshake“ ist für Benutzer zu kompliziert und der Tab in der Navigation sollte entsprechend umbenannt werden.
- **Projekt-Engine:** Die Projekt-Engine (siehe *Engine*) sollte beim Hover über einen der Kreise einen Tooltip anzeigen.
- **Hyperlink auf Seite „Bildung“:** Der Hyperlink, welcher auf der Seite Bildung auf die Module verweist, ist nicht gut genug erkennbar. Alle drei Testpersonen hatten Schwierigkeiten diesen zu finden. Der Aufbau der

Seite sollte entsprechend angepasst werden und die Verlinkung sollte besser ersichtlich platziert werden.

- **Viel Text:** Auf einigen Seiten ist noch viel Text vorhanden. Wie im Kapitel *Inhalt erstellen* beschrieben, wurden viele Seiten aus dem Prototypen oder HSLU-Webseite übernommen. Eine Untermalung diese Informationen mit Bild und Video wäre empfehlenswert.
-

Als Ganzes betrachtet waren die Testpersonen mit dem Design der Webseite zufrieden und abgesehen von den oben erwähnten Problemen stechen keine Features der Webseite als besonders schlecht heraus.

6.4 Testing mit Lighthouse

Die Performance, Accessibility und Search Engine Optimization der Webseite wurde mit Lighthouse[34] gemessen.
Lighthouse ist ein Open-Source Tool, welche eine Webseite einfach im Browser auf Performance, Accessibility und SEO überprüft werden kann.
Eine erste Durchführung der Lighthouse Test auf der Empfehlungs-System Seite ergab einen Accessibility-Score von nur 78.

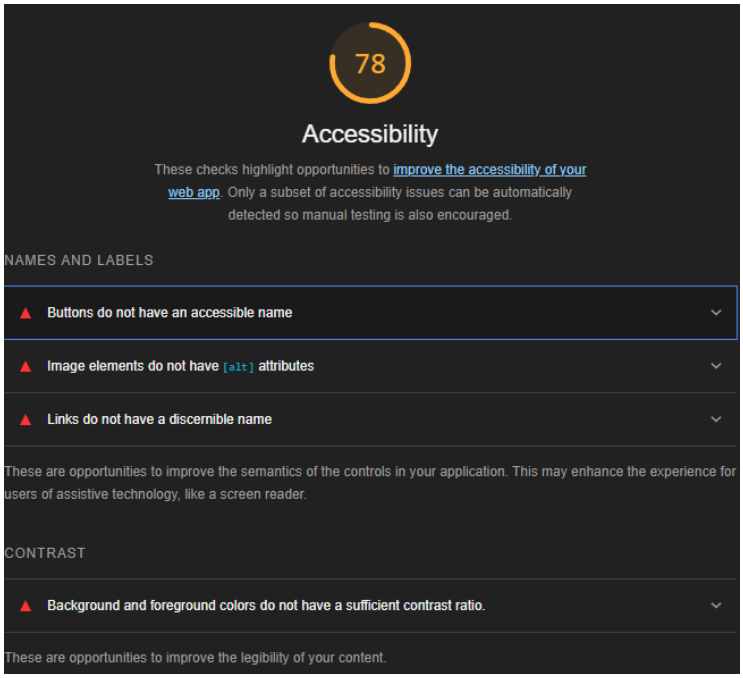


Abbildung 43: Erster Lighthouse durchlauf

Lighthouse bewertet die Seite jedoch nicht nur mit einem Score, sondern es gibt auch direkt Hinweise, wie man diesen Score beziehungsweise die Accessibility der Webseite verbessern könnte. So wird zum Beispiel verlangt, dass alle Links mit dem ***`aria-label`*** Property erweitert werden und das die Image-Element auf der Webseite alle einen alt Property benötigen. Durch das Beheben dieser Probleme konnte der Accessibility-Score auf 97 erhöht werden.

Die Lighthouse Test wurde für die folgenden Seiten durchgeführt und die Erreichte Punktezahl ist jeweils in der Tabelle zu sehen. Die von Lighthouse generierten Reports sind im Anhang unter *Lighthouse* zu finden.

Tabelle 9: Performance der Webseite

Seite	Performance	Accessibility	SEO	Bemerkung
Empfehlungs-System	90	97	92	
Forschungsprojekt Übersicht	80	98	92	Performance durch Thumbnails der Forschungsprojekte vermindert.
Methoden-Ansicht (Methode Diary Studies)	96	96	92	
Mitarbeitenden - Ansicht (Mitarbeitender: Jan Kalbermatter)	84	98	92	Performance durch Thumbnails der

[34] https://developer.chrome.com/docs/lighthouse/overview/

				Forschungsprojekte vermindert.

Wie in den Anforderungen (siehe Anhang *Anforderungen*) vorgeben wurden folgenden Punktezahlen als Ziel gesetzt.

Performance
- Optimale: Eine Punktezahl grösser/gleich 90
- Akzeptiert: Eine Punktezahl grösser/gleich 80

Accessibility
- Optimale: Eine Punktezahl grösser/gleich 90
- Akzeptiert: Eine Punktezahl grösser/gleich 80

SEO
- Optimale: Eine Punktezahl grösser/gleich 80
- Akzeptiert: Eine Punktezahl grösser/gleich 70

Die Performance-Scores konnte dank SSG und ohne Client-Side-Rendering erreicht werden. Bilder welche die ***`next/image`*** Komponente des NextJS Frameworks verwenden, werden automatisch Lazy Loaded[35]. Lazy Loading bedeutet, dass die Bilder erst dann geladen werden, wenn sie in den Viewport des Benutzers fallen. Um die Ladezeit der Bilder weiter zu verbessern könnt deren Grösse im CMS beispielsweise limitiert werden.

Um die Accessibility-Scores zu erreichen, mussten die Lighthouse-Test mehrere Male durchgeführt und entsprechende Anpassungen am Code gemacht werden. Wie zu Beginn des Kapitels beschrieben waren die Scores zuerst zu tief, weshalb vor allem Bilder und Links Elemente mit beschreibenden Properties (***`aria-label`***, ***`alt`***) ergänzt werden mussten.
Durch diese Anpassungen fielen die Bewertung anschliessend sehr gut aus.

Um den SEO-Score zu erreichen, musste, während dem Projekt gegen die eigentliche Erwartungen keine besonderen Vorkehrungen vorgenommen werden. Der SEO-Score kann erhöht werden, wenn sich eine Webseite an HTML-Konventionen wie die Verwendung eines HTML-DOCTYPE. Aber auch das Verbinden mit der Webseite über HTTPS anstelle von HTTP hat positiven Einfluss auf die Bewertung. NextJS als Framework hat beim Erreichen des SEO-Scores sicherlich auch beigetragen. Ein wichtiger Punkt für SEO ist, dass die Inhalts-Daten und Metadaten einer Seite beim Laden der Seite bereits existieren. Hier kann also SSG eingesetzt werden, um Seiten auf der Server-Seite zu genieren und diese dem Benutzer samt Inhalt und Metadaten zur Verfügung zu stellen.[36]

[35] https://nextjs.org/docs/api-reference/next/image
[36] https://nextjs.org/learn/seo/rendering-and-ranking/rendering-strategies

6.5 Responsive Design

Durch den Einsatz von TailwindCSS (Siehe Kapitel *TailwindCSS*) wurde das Problem des Responsive Design grösstenteils gelöst und grosse Anpassungen konnten vermieden werden.
Die Validierung der Responsive Design erfolgte mithilfe der Google Chrome Entwickler-Tools. Hier können Webseiten in den genauen Dimensionen von Mobilen Geräten (wie z.B. iPhone 12 oder Pixel 5) oder von Tablets (z.B. iPad Air) angezeigt werden.

Durch den Komponente-basierten Aufbau von NextJS ist es nicht nötig, jede Seite auf ihre Responsiveness zu testen, sondern es reicht, die Responsiveness der einzelnen Komponenten sicherzustellen.
Folgend ein Beispiel wie der Inhalt der Webseite responsiv Dargestellt wird anhand des *Empfehlungs-System*:

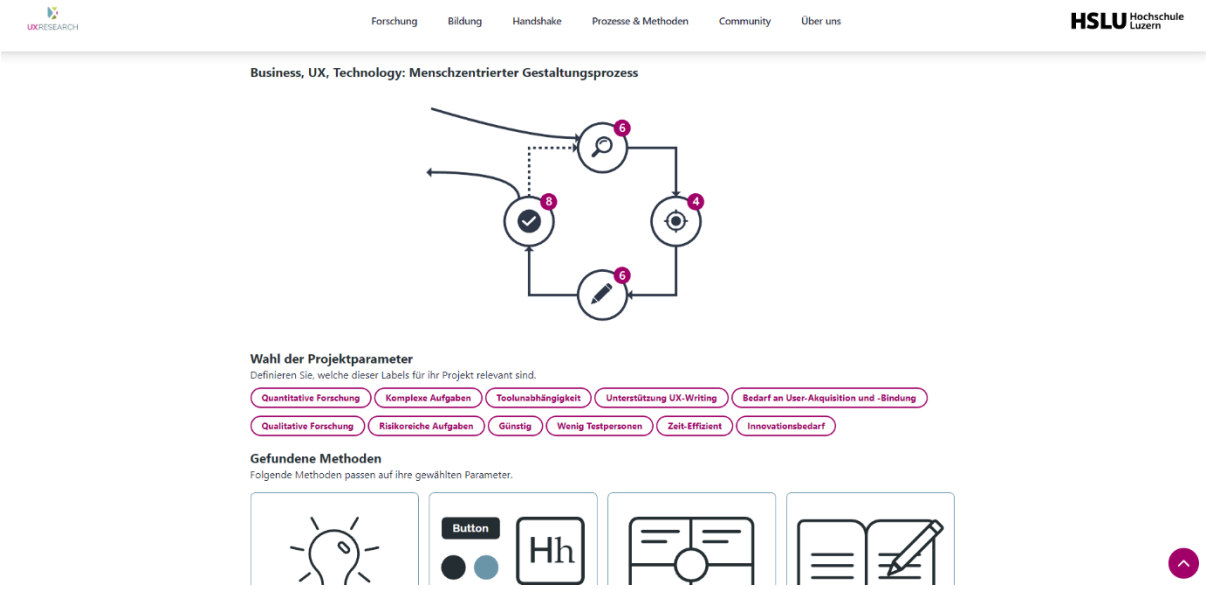


Abbildung 44: Desktop-Ansicht (1080p)

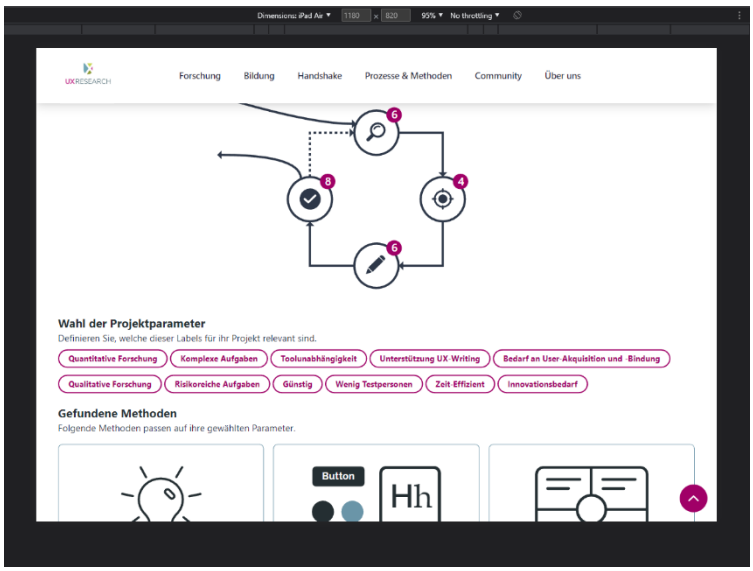


Abbildung 45: Tablet-Ansicht (iPad Air)

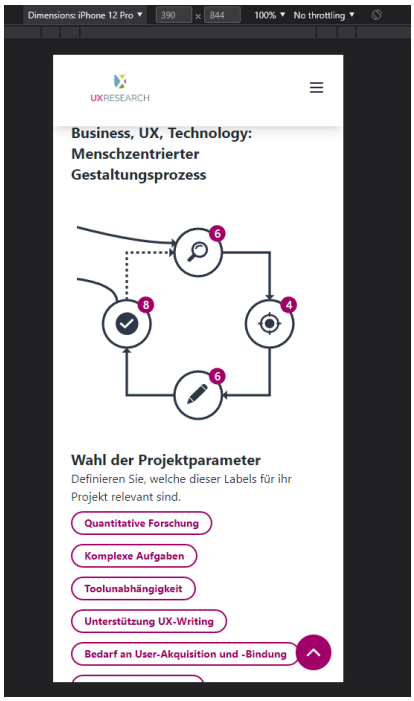


Abbildung 46: Mobile-Ansicht 1 (iPhone 12)

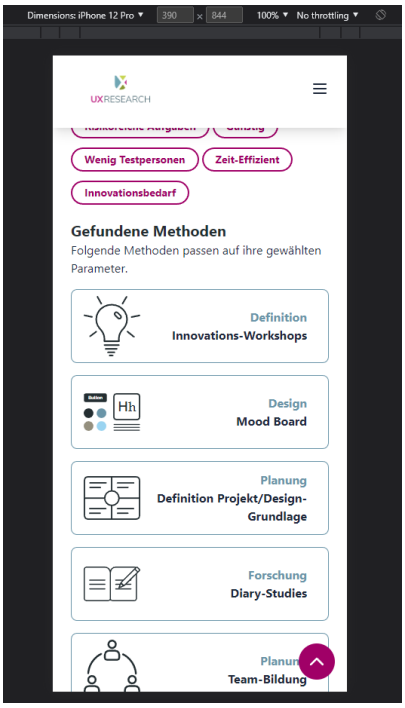


Abbildung 47: Mobile-Ansicht 2 (iPhone 12)

Wie in Abbildung 46 und Abbldung 45 zu sehen, nehmen die Filter in der Mobilen-Ansicht viel platz in Anspruch. Das ist ein bereich welche vielleicht noch verbessert werden könne, durch Anzeigen der Filter in einem Modal. Für die Mobile Ansicht wurde bereits die Methoden-Card angepasst, damit der Benutzer nicht zu weit Scrollen muss, um eine Übersicht über diese zu erhalten.

6.6 Funktionale Anforderungen

Auf die Funktionalen Anforderungen wird mittels ihrer Nummer verwiesen und die Erfüllung der Anforderung wird untenstehend definiert. Die Funktionalen Anforderungen können im Anhang unter *Anforderungen* gefunden werden.
In den Bemerkungen wird auf die jeweiligen Kapitel verwiesen, welche den Status beweisen.

Tabelle 10: Evaluierung der Funktionale Anforderungen

#	Status	Bemerkung	Beweis
1.1	Erfüllt		Siehe Empfehlungs-System
1.2	Erfüllt		Siehe Inhalt erstellen
1.3	Erfüllt		Siehe Inhalt erstellen und Gefundene Methoden
1.4	Erfüllt		Siehe E-Mail versenden
1.5	Teilweise erfüllt	Das Erstellen der Entsprechenden Seiten wird der UX Research Group überlassen	Siehe Kompetenzen vermitteln und Ausblick

2.1	Erfüllt		Siehe CMS Handbuch
2.2	Erfüllt		Siehe Prismic

6.7 Nicht-Funktionale Anforderungen

Auf die Nicht-Funktionalen Anforderungen wird mittels ihrer Nummer verwiesen und die Erfüllung der Anforderung werden untenstehend definiert. Die Nicht-Funktionalen Anforderungen können im Anhang unter *Anforderungen* gefunden werden.
In den Bemerkungen wird auf die jeweiligen Kapitel verwiesen, welche den Status beweisen.

Tabelle 11: Evaluierung der Nicht-Funktionale Anforderungen

#	Status	Bemerkung	Beweis
1.1	Erfüllt		Siehe Testing mit Lighthouse
1.2	Erfüllt		Siehe Testing mit Lighthouse
1.3	Erfüllt		Siehe Testing mit Lighthouse
2.1	Grösstenteils erfüllt	Einige Probleme mit der Navigation wurden im Rahmen der Usability-Tests festgestellt.	Siehe Usability-Tests
2.2	Erfüllt		Siehe Responsive Design
2.3	Erfüllt		Siehe Usability Tests
3.1	Erfüllt		Siehe Validierung und zweite Runde
3.2	Erfüllt		Siehe Anhang CMS Handbuch

7 Ausblick

7.1 Persönliches Fazit

Ich bin froh, konnte ich das Projekt erfolgreich abschliessen und bin zufrieden mit dem erarbeiteten Web-Auftritt.

Es gibt noch einige Erweiterungsmöglichkeiten und es sind wahrscheinlich einige Woche Arbeit nötig, bis der Web-Auftritt wirklich allen Erwartungen und Anforderungen der UX Research Group entspricht. Es war jedoch von Anfang an klar, dass auf diese Arbeit aufgebaut werden soll und ich bin zufrieden mit dem Grundstein, welchen ich in der Umsetzung mit dieser Arbeit gelegt habe.

Aufgrund von Krankheit sind gegen Ende des Projekts eine Woche an Arbeitszeit verloren gegangen. Nach Nachfrage Meier René (Modulverantwortlicher BAA) konnte das Projekt um drei Tage verlängert werden, was für den sauberen Abschluss des Projekt geholfen hat.

Die Workshops mit der UX Research Group und den potenziellen Benutzer waren sehr interessant und Wertvolle Inputs konnte daraus gewonnen werden. Die Umsetzung solche Workshops war für mich neu, jedoch konnte mir hier die UX Research Group gut unter die Arme greifen.

Das Arbeiten zusammen mit den Mitgliedern der Forschungsgruppe habe ich über das gesamte Projekt als angenehm empfunden und es war sehr hilfreichen, einen Auftraggeber zu haben welcher so viele Ideen und Verbesserungsvorschläge zum Projekt einbringt.

Die Auswahl des CMS stellte sich als Herausforderung heraus, da diese auf die Bedürfnissen und Skills aller Mitarbeitenden der Forschungsgruppe ausgerichtet werden musst und es so viele verschiedene Angebote für CMS-Lösungen gab.

Zudem war die Einarbeitung in NextJS nicht einfach. Obwohl bereits Berufserfahrung mit JavaScript und Erfahrung mit modernen Web-Technologien aus dem Modul „WebLab“ vorhanden waren, konnte ein effektives Verständnis für die Arbeit mit dem Framework konnte erst durch das effektive Arbeiten damit erreicht werden. Das persönliche Interesse mit NextJS zu arbeiten war jedoch sehr Gross und der Lernprozess wurden durch die ausführliche Dokumentation von NextJS erleichtert. die gelernten Konzepte und Arbeitsweise werden mit auch noch in Zukunft Mehrwert bieten.

7.2 Erweiterungsmöglichkeiten

Die im Rahmen dieses Projektes erstellte Webseite bildet eine Grundlage, auf welcher zukünftige Weiterentwicklungen möglich sind. Einige Erweiterungsmöglichkeiten wurden mittels Workshops und Interviews von Teilnehmern eigenständig eingebracht. Bei anderen handelt es sich um eigene Ideen des Autoren wie man die Web-Applikation noch verbessern könnte.

Einige Erweiterungsmöglichkeiten werden folgend erläutert.

7.2.1 Kalender

Der Kalender (siehe Kapitel Kalender) könnte noch übersichtlicher gestaltet werden. In den Workshops und Interview stachen vor allem zwei Idee als potenzielle Verbesserungen des Kalenders heraus

- **Datepicker mit einer Range:** Die beiden Datepicker könnten zu einem einzelnen Element verbunden werden, in welchem das Start- und das Enddatum ausgewählt werden können.
- **Events im Datepicker markieren:** Events könnten durch eine farbige Markierung direkt im Datepicker gekennzeichnet werden. Dadurch wird dem Benutzer schon mitgeteilt, ob er mit seiner Suche ein Event finden oder ob er keine Resultate zurückerhalten wird.

7.2.2 Kontaktform

Die Kontaktform benutzt momentan die im Projekt erstellte Gmail-Adresse. Diese könnte durch die von Marcel Uhr erstellte E-Mail ux-research@hslu.ch ersetzt werden. Hierzu müsste jedoch zuerst auf Seite der HSLU abgeklärt werden, zu welchem SMTP-Server eine Verbindung aufgebaut werden muss und ob diese technische Umsetzung überhaupt so erlaubt ist.
Diese Abklärung fand im Rahmen des Projekts nicht mehr statt.

7.2.3 Neue Slices

Neue Slices bieten den Editoren zum CMS mehr Möglichkeiten, den Inhalt der Webseite interessant und verständlich zu gestalten. Im Rahmen dieses Projekts wurden einige Slices erstellt oder aus dem Starterkit übernommen. Mit diesen Slices können bereits simple Seiten erstellt werden, es besteht jedoch immer die Möglichkeit, mit Slicemachine neue Slices zu erstellen. Folgenden einige potenzielle Slice welche noch erstellte werden könnten:

- **Accordion:** Mit einem Accordion kann Text vom Benutzer selbst ein- und ausgeblendet werden. In den Workshops (siehe *Feedback Workshop*) wurde das Feedback mitgeteilt, dass es sehr Textlastige Seiten gibt, auf welchen die es schwer fallen kann die Übersicht zu behalten. Ein Accordion könnte soll ein Problem lösen.
  Siehe Chakra UI - Accordion für ein Beispiel
- **Kommentare:** Blogeinträge könnten von einer Kommentar-Funktion profitieren, in welcher sich Benutzer über die Artikel austauschen können. Laut Prismic selbst gibt es keine Pläne, eine Write API (Inhalt für Prismic über

das Frontend erstellen) zu kreieren[37]. Weshalb eine Kommentar-Funktion nicht bereits umgesetzt wurde. Um Kommentare zu speichern würde es ein weitere Backend benötigen, beziehungsweise eine eigene API inklusive Datenbank welche die Kommentar-Daten speichert.

- **Login/Register/Profil:** Angemeldete Benutzer sind ein Muss für Features wie das Schreiben von Kommentaren, damit nicht jeder Benutzer schreiben kann, was er will und die Besitzer der Webseite mehr Kontrolle über inakzeptables Nutzerverhalten haben. Ein Login ermöglich es auch, dass Benutzer sich selber präsentieren in eine Profil präsentieren können und der Community Aspekt der Seite kann verbessert werden. Auch hier würde es ein eigenes Backend für die Benutzer-Anmeldung brauchen.
- **Archivierte/Nicht-Archivierte Blogeinträge:** Die Aufteilung der Blogeinträge in eine Ansicht der Archivierten sowie eine der Aktiven wurde im Rahmen eines Workshops auch noch vorgeschlagen. Das Abfragen von archivierten Dokumenten über die Prismic API ist nicht möglich. Es wäre jedoch vorstellbar, die Archivierung nicht über Prismic selber, sondern über einen Toggle in den Standard-Daten der Blogs zu regeln. Im Prismic Forum wird zudem ein Workaround mit „Releases" erwähnt, welche hier auch als potenzielle Lösung möglich wäre[38].

7.2.4 Inhalt aufnehmen

Die zu vermittelnden Kompetenzen der UX Research Group und die Inhalte gemäss Module und dem Studiengang Digital Ideation wurden nicht im Rahmen dieses Projekts von den Autoren aufgenommen. Es liegt an den Mitarbeitenden der UX Research Group oder an anderen CMS-Editoren, diesen Inhalt zu erstellen und zu verwalten.

Die erstellte Webseite bietet hier lediglich ein Framework an, die eigenen Ideen und Konzept noch umzusetzen und wie gewünscht darzustellen.

7.2.5 Empfehlungs-System

Das Empfehlungs-System könnten an einigen Stellen noch erweitert, oder es könnten weitere interaktive Hilfe-Tools erstellt werden.
Durch das Erstellen einer Login-Komponente und der dadurch ermöglichten Registrierung und Anmeldung von Benutzern könnten eingegebene Daten im Empfehlungs-System zu persistiert werden. Auch ohne Login-Komponente wäre es möglich, einen PDF-Export zu erstellen, welcher die vom Empfehlungs-System erstellten Übersichten darstellen.
Es wäre möglich ein Template für gewählte Methoden passend zu den eingegeben Projekt-Parametern erstellen zu lassen.
Die spezifische Umsetzung dieser Features müsste analysiert werden.

7.2.6 Design der Seite

Das Design der Seite könnten beliebig angepasst werden. Aus den Workshops (siehe *Feedback Workshop*) kam hier auch die Beschwerde auf, dass die Seite zu wenige auf Farben setzt. Durch Farben und Animationen, wie zum Beispiel den

[37] https://community.prismic.io/t/write-api-save-data-to-prismic-from-your-app/283/3
[38] https://community.prismic.io/t/query-unpublished-draft-archive-documents/6537

Magenta Elementen und dem Zoom-In der Bilder (siehe Kapitel *Forschungsprojekte finden*), könnte die Webseite viel dynamischer/lebhafter gestaltet werden.
Zudem empfiehlt es sich, der Text-Inhalt auf den existieren Seiten noch mehr mit Video und Bild zu untermalen.

7.2.7 Weiter Community Features

Es könnten weitere Community Features, wie die bestehenden Events und Blogs, hinzugefügt werden.

- Eine Idee im Rahmen des ersten Workshops war es, über die Webseite Zoom UX Workshops anzubieten. Hierzu könnte man dem Benutzer die Möglichkeit bieten, sich für solche Kurse anzumelden.
- In den Workshops (siehe *Feedback Workshop*) wurde derselbe Vorschlag auch bezüglich den Events geäussert. Hier fänden es Benutzer gut, wenn sie sich auf Events anmelden oder Events in den eigenen Kalender hinzufügen könnten.

8 Anhänge

Die aufgelisteten Anhänge sind als separate Dokument im Ordner Anhang.zip zu finden, welcher der Arbeit beiliegt. Die Ordnerstruktur entspricht der folgenden Auflistung:

8.1 Aufgabenstellung

8.2 Anforderungen

8.3 Rahmenplan

8.4 Risikomanagement

8.5 Sprintplanung

8.6 CMS-Handbuch

8.7 Testing

8.7.1 Usability-Tests

8.7.2 Blackbox-Tests

8.8 Wireframes

8.9 Lighthouse

8.9.1 Empfehlungs-System

8.9.2 Forschungsprojekte-Übersicht

8.9.3 Methoden-Ansicht

8.9.4 Mitarbeitenden-Ansicht

8.10 CMS-Auswahl

8.10.1 Erste Runde

8.10.2 Zweite Runde

8.11 Workshop - Bilder

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

11 Literaturverzeichnis

Hamilton, T. (5. November 2022). *Guru99*. Von What is BLACK Box Testing?: https://www.guru99.com/black-box-testing.html abgerufen

Moran, K. (1. Dezember 2019). *nngroup*. Von Usability Testing 101: https://www.nngroup.com/articles/usability-testing-101/ abgerufen

NextJS. (30. October 2022). Von Documentation: https://nextjs.org/docs abgerufen

NextJS. (8. Oktober 2022). Von NextJS - Documentation React Hydration Error: https://nextjs.org/docs/messages/react-hydration-error abgerufen

NodeMailer. (5. Dezember 2022). *NodeMailer*. Von NodeMailer Dokumentation - Using Gmail: https://nodemailer.com/usage/using-gmail/ abgerufen

Pavlutin, D. (29. Mai 2022). *dmitripavlutin.com*. Von dmitripavlutin - A Simple Explanation of React.useEffect(): https://dmitripavlutin.com/react-useeffect-explanation/ abgerufen

Prismic. (5. Oktober 2022). Von Prismic Crash Course: https://prismic.io/docs/nextjs abgerufen

React. (6. Februar 2019). Von React Documentation - Hooks: https://reactjs.org/docs/hooks-state.html abgerufen

Wikipedia. (3. November 2022). Von Wikipedia - Webhook: https://en.wikipedia.org/wiki/Webhook abgerufen

Wikipedia. (14. November 2022). *Wikipedi*. Von Wikipedia - Hydration: https://en.wikipedia.org/wiki/Hydration_(web_development) abgerufen

Wikipedia. (15. September 2022). *Wikipedia*. Von Black-Box Testing: https://de.wikipedia.org/wiki/Black-Box-Test abgerufen

Zanas, E.-A. (8. Dezember 2022). *Strapi*. Von Strapi - Traditional vs Headless CMS: A Comparision: https://strapi.io/blog/traditional-vs-headless-cms-a-comparison abgerufen