Kolibri

IP 5 Ramona Marti, Lea Burki

Auswahlkomponente für das WebUI Toolkit "Kolibri"

Informatik Projekt 5

Lea Burki und Ramona Marti

Abstract

Formulare ausfüllen, besonders Dropdown-Menüs, kann oft unübersichtlich sein. Um dies zu verbessern, entsteht im Kolibri Toolkit eine neue, benutzerfreundlichere Auswahlkomponente. Das Ziel ist, eine anwenderfreundliche und intuitive Schnittstelle für Webanwender zu schaffen. Gleichzeitig ermöglicht sie für Entwickler eine leichte Einbindung und effiziente Verwaltung. Der Fokus dieses Projekts liegt speziell auf einer Länderauswahl-Komponente. Mit Prototyping in der Designentwicklung und Refactoring bei der Implementation entsteht eine von Anwendern bereits getestete Komponente. Die resultierende Lösung dient als Erweiterung des Toolkits Kolibri. Durch ihren modularen Aufbau ist die einfache Benutzung und Erweiterung gewährleistet. Eine nachfolgende Weiterentwicklung bietet die Möglichkeit der Generalisierung und Optimierung, als auch der Ergänzung von Zusatzfunktionen.

Inhaltsverzeichnis

1 Einleitung

Dieses Projekt setzt sich mit der Herausforderung auseinander, intuitive und zugleich technisch fortschrittliche Web-Komponenten zu entwickeln. Die Auswahlkomponente ist ein Beispiel dafür, wie komplexe Benutzeranforderungen mit einfacher, aber leistungsfähiger Technologie erfüllt werden kann. Sie zeigt auf, wie das Kolibri Toolkit nicht nur als Sammlung von Tools, sondern als integriertes System zur Verbesserung der Webentwicklung beiträgt.

1.1 Problemstellung

Ein zentrales Problem, das in diesem Projekt adressiert wird, ist die Unzulänglichkeit der standardisierten Auswahlkomponenten im Web – insbesondere die select und datalist Elemente. Diese weisen Mängel in Design, Integration in einheitliche Datenmodelle, Konsistenz im Interaktionsdesign und Benutzereffizienz auf. Beispielsweise wenn grosse Datenmengen dargestellt werden müssen, kann die unzureichende Gestaltung von select Elementen zu einer verwirrenden Benutzererfahrung führen. Das nachfolgende Kapitel 1.2 basiert auf den hier genannten Problemen.

1.2 Ziel

Das primäre Ziel dieser Arbeit ist die Gestaltung und Implementierung einer Dropdown-Auswahlkomponente. Diese ermöglicht eine effiziente und ansprechende Auswahl aus mehreren vorgegebenen Optionen. Einerseits umfasst dieses Projekt die Synchronisation von Design und Implementation. Weiter beinhaltet die Komponente die Integration in das Kolibri-Designsystem. Die Auswertung von Benutzertests hilft beim Visualisieren der Inhalte sowie dem Interaktionsdesign. Automatisierte Umsetzungstests begünstigen einen fehlerfreien Code. Ebenfalls beinhaltet die Erstellung dieser Komponente die Anbindung an Kolibri-Modelle, -Controller und -Projektoren sowie die Durchführung von Usability-Tests. Spezifische Kriterien für die Bewertung des Erfolgs der Komponente, einschliesslich Benutzerzufriedenheit und Integrationseffizienz, nehmen ihren Platz ein. Dabei gilt zu beachten, dass sich die Ziele nicht ausserhalb des definierten Projekts bewegen.

1.3 Out of Scope

Dieses Projekt konzentriert sich ausschliesslich auf die Entwicklung der Auswahlkomponente und deren Integration in das bestehende Kolibri-Framework. Aspekte ausserhalb dieses Rahmens, wie die Überarbeitung anderer UI-Komponenten oder die Erweiterung der Kern-Codebasis von Kolibri, fallen nicht in den Umfang dieser Arbeit. Dadurch wird sichergestellt, dass die Arbeit fokussiert und zielgerichtet bleibt.

Der Fokus für das Projekt liegt spezifisch auf der Entwicklung einer Auswahlkomponente für Länder. Dies bedeutet, dass andere potenzielle Anwendungsbereiche wie die Integration von Datums- oder Jahreszahl-Auswahl keine Berücksichtigung finden. Dieser begrenzte Fokus ermöglicht, eine massgeschneiderte Lösung zu entwickeln. Diese ist speziell auf die Anforderungen und Herausforderungen der Länderauswahl zugeschnitten. Dadurch gelingt die Sicherstellung, dass die vorhandenen Ressourcen gezielt für die Optimierung der Benutzererfahrung und Funktionalität im Kontext der Länderauswahl einsetzbar sind.

1.4 Leitfaden

Dieser Bericht gliedert sich in die Teile Recherche, Design und Implementation. Das Konzeptionelle findet sich in den Kapiteln 2 bis 3.2 wieder. Der Abschnitt 3.3 prüft das erarbeitete Konzept. Dieses findet in nachfolgendem Unterkapitel 3.4 seine Umsetzung. Das Resultat des Designs als auch der Implementation ist unter der Nummer 4 Länder-Auswahlkomponente beschrieben. Kapitel 5 bewertet das Produkt und die erhaltenen Erkenntnisse.

2 Grundlagen

Die Grundlagen ermöglichen einen Einblick in den Hintergrund des Projekts. Die vorhandene Basis findet sich in der Beschreibung der Ausgangslage des Kolibri wieder. Weiter setzt es den Grundstein für das Verständnis und Anwendung der Auswahlkomponente.

2.1 Ausgangslage

Die FHNW betreibt und verwaltet das Toolkit Kolibri. Dabei wird das Projekt unter anderem durch Resultate aus Web-Projekten der Hochschule für Technik schrittweise weiterentwickelt. Interessierte Entwickler aus aller Welt können die Open-Source Codebasis ohne Probleme verwenden. Die Implementation der standardisierten Komponenten ist in reinem JavaScript umgesetzt. Daher sind keine komplexen Management-Systeme wie npm von Nöten. Die intuitive Zugänglichkeit von Kolibri ermöglicht einen schnellen Einstieg ohne lange Einarbeitungszeiten oder komplexe Tutorials. Auf der anderen Seite sind jedoch noch nicht alle Interaktionen vollständig erhältlich.

Um die Auswahlkomponente einfacher einbinden zu können, wurde aus der Github-Codebasis von Kolibri (Appendix B) vom 18.09.2023 ein Fork erstellt. Dieser Stand enthält unter anderem bereits die SimpleInput Komponenten, welche alle Felder des input Tags verwalten kann. Noch nicht eingebundene Komponenten früherer Projekte sind unter dem Ordner contrib bereitgestellt.

Im Design Tool Figma existiert bereits ein Designsystem, als auch in das Toolkit eingebundene UI-Komponenten. Ein Satz von Icons ist bereits vorhanden, welcher bei Bedarf erweitert werden kann. Die bereits vorhandenen Elemente besitzen einen Komponenten-Aufbau. Dies erleichtert die Anwendung der Bausteine in der Entwicklung weiterer Elemente.

Ein gemeinsames Verständnis der Auswahl-Komponente und deren Herausforderung stellt eine wichtige Basis dar. Um diese Übereinstimmung zu schaffen, folgt die detaillierte Beschreibung in nachfolgenden Kapitel.

2.2 Auswahlkomponente

Folgende Abschnitte beschreiben das gemeinsame Verständnis einer Auswahlkomponente, als auch die Wichtigkeit in der Anwendung, im besonderen mit grossen Datenmengen. Es zeigt die Herausforderungen und erste Lösungsgedanken auf.

2.2.1 Definition und Bedeutung

Eine Auswahlkomponente, oft ein integraler Bestandteil moderner Benutzeroberflächen, ist ein interaktives Element in der Softwareentwicklung. Es ermöglicht den Benutzern, eine oder mehrere Optionen aus einer vordefinierten Liste auszuwählen. Ihre Anwendung reicht von einfachen Dropdown-Menüs bis hin zu komplexen Such- und Filtermechanismen. In der wissenschaftlichen Terminologie wird eine Auswahlkomponente oft als eine Schnittstelle definiert. Diese ermöglicht eine effektive Interaktion zwischen dem Benutzer und den verfügbaren Datenoptionen.

2.2.2 Wichtigkeit bei umfangreichen Datensätzen

(UX PICKLE, 2024) Die Bedeutung von Auswahlkomponenten nimmt zu, insbesondere in Szenarien, in denen Benutzer aus einer grossen Menge von Daten auswählen müssen. Ein klassisches Beispiel hierfür ist die Auswahl eines Wohnsitzlandes aus allen Ländern der Welt. In solchen Fällen muss die Auswahlkomponente nicht nur eine umfassende Liste der verfügbaren Optionen bereitstellen, sondern auch eine benutzerfreundliche Schnittstelle, um die Suche und Auswahl zu erleichtern.

2.2.3 Herausforderungen und Lösungsansätze

(Maurer Spence, 2024) Die Hauptherausforderung bei der Gestaltung einer Auswahlkomponente für grosse Datensätze liegt in der Bereitstellung einer nahtlosen und intuitiven Benutzererfahrung. Zu viele Optionen können überfordernd wirken und die Benutzerfreundlichkeit beeinträchtigen. Um dies zu bewältigen, kommen oft fortschrittliche Techniken wie inkrementelle Suche, Filterung und Kategorisierung zum Einsatz.

Inkrementelle Suchmechanismen ermöglichen es den Benutzern, ihre Auswahl durch Eingabe von Suchbegriffen zu verfeinern, wodurch die Liste der Optionen dynamisch gefiltert wird. Dies ist besonders effektiv, wenn die Auswahl aus einer umfangreichen Liste – wie der aller Länder der Welt – erfolgen soll. Durch die Filterung nach bestimmten Kriterien, wie etwa dem Kontinent, können Anwender die Auswahl effizienter eingrenzen.

2.2.4 Fazit

Zusammenfassend lässt sich sagen, dass Auswahlkomponenten eine wichtige Rolle in der Gestaltung benutzerzentrierter Software spielen, insbesondere wenn es um die Navigation und Auswahl aus umfangreichen Datensätzen geht. Die Herausforderung besteht darin, eine Balance zwischen der Bereitstellung aller notwendigen Informationen und der Gewährleistung einer intuitiven, effizienten Benutzeroberfläche zu finden. Innovative Ansätze in der UI/UX-Designphilosophie, wie die Implementierung der inkrementellen Suche und intelligenten Filtermechanismen, sind dabei unerlässlich, um eine optimale Benutzererfahrung zu ermöglichen. Zusätzlich kann das Analysieren der Fehler von Konkurrenzprodukten hilfreich sein.

2.3 Konkurrenzanalyse und Positionierung

Im Rahmen der Entwicklung der Kolibri Dropdown-Komponente ist die Untersuchung der bestehenden Lösungen ein wichtiger Bestandteil. Dadurch zeigen sich Stärken und Schwächen gängiger Auswahlkomponenten auf. Diese Analyse dient als Grundlage für die Gestaltung einer benutzerfreundlichen, technisch effizienten und unabhängigen Komponente. Diese ist speziell auf die Bedürfnisse Endnutzer zugeschnitten.

2.3.1 Vergleich bestehender Lösungen

Standard-HTML-Elemente wie <select> und <datalist> bieten zwar grundlegende Funktionalitäten, sind jedoch in Bezug auf Designflexibilität und Benutzerfreundlichkeit limitiert. Bibliotheksabhängige Lösungen wie Bootstrap, React und Vue.js erweitern zwar die Möglichkeiten, bringen aber zusätzliche Abhängigkeiten und Einarbeitungsaufwand mit sich. jQuery-basierte Lösungen bieten eine verbesserte Benutzererfahrung, sind jedoch ebenfalls von externen Bibliotheken abhängig.

(W3Schools, 2023) Das select-Tag von HTML bietet nur sehr begrenzte Möglichkeiten in der Gestaltung bzw. Darstellung der Auswahl-Elemente. Grosse Datenmengen benutzerfreundlich darzustellen stellt sich als sehr schwierig dar. Das HTML-Element wird von allen Browsern unterstützt und nimmt nur vorgegebene Werte an.
(W3Schools, 2023) Eine HTML-datalist ist an ein Input-Feld gebunden und bietet ebenfalls fast kein Spielraum für ein eigenes Design. Dabei können eigene Werte eingegeben werden, was nicht in allen Fällen erwünscht ist. Ein Vorteil ist, die Liste filtert bzw. passt sich bei einer Eingabe automatisch an.

2.3.2 Vergleichende Übersicht der Dropdown-Komponenten

Tabelle 2.1 - Vergleich der Konkurrenz Produkte

Komponente	Vorteile	Nachteile	Besonderheiten
HTML select	Browserübergreifende Unterstützung	Begrenzte Gestaltungsmöglichkeiten	Einfach, nimmt nur vorgegebene Werte an
HTML datalist	Automatische Filterung bei Eingabe	Begrenztes Design, ermöglicht eigene	Bindung an ein Input-Feld
use-bootstrap-select	Umfangreiche Gestaltungsmöglichkeiten	Erfordert Einarbeitung in die Bibliothek	Unterstützt Suchfunktion, Mehrfachauswahl
EasyDropDown.js	Leichtgewichtig, anpassbar	Dokumentation notwendig	Drei Designs verfügbar
React Dropdown	Unterstützt Optgroups,	Abhängig von React	Einfache Installation mit npm
React Bootstrap Dropdown	Umfangreiche Anpassungsoptionen	Erfordert mehrere Abhängigkeiten	Unterstützt verschiedene Grössen und
Vue-Multiselect	Multi-Selektion, Tagging	Abhängig von Vue.js	Ermöglicht benutzerdefinierte Vorlagen
Chosen (jQuery-basiert)	Suchfunktion, Multi-Selektion	Abhängig von jQuery	Anpassbares Erscheinungsbild
Kolibri Dropdown	Unabhängig, benutzerfreundlich	Kolibri Toolkit spezifisch programmiert	Einfache Integration, anpassbar, hohe Performance

2.3.3 Spezifische Herausforderungen und Einschränkungen

Diese Lösungen stellen Entwickler oft vor Herausforderungen hinsichtlich der Integration, Anpassungsfähigkeit und Performance. Insbesondere bei grossen Datenmengen und spezifischen Designanforderungen offenbaren sich die Grenzen herkömmlicher Dropdown-Elemente.

2.3.4 Eigene Entwicklung: Kolibri Dropdown-Komponente

Die Kolibri Dropdown-Komponente soll unabhängig von externen Bibliotheken funktionieren. Diese Komponente zeichnet sich durch eine hohe Flexibilität, einfache Integration und benutzerfreundliche Interaktion aus. Sie folgt den Design- und Codierungsstandards von Kolibri und sie soll einfach in verschiedene Webprojekte integrierbar sein.

2.3.5 Vorteile der Kolibri Dropdown-Komponente

Unabhängigkeit von Drittbibliotheken:
Die neue Komponente benötigt keine externen Abhängigkeiten, was die Sicherheit und Stabilität erhöht.

Benutzerfreundlichkeit und Zugänglichkeit:
Die Komponente entwickelt sich mit Fokus auf eine intuitive und barrierefreie Benutzererfahrung.

Anpassungsfähigkeit und Erweiterbarkeit:
Sie lässt sich leicht an unterschiedliche Designrichtlinien und Anforderungen anpassen.
Technische Effizienz:
Durch die Vermeidung unnötiger Bibliotheken wird eine hohe Performance erreicht.

2.3.5 Fazit

Die Kolibri Dropdown-Komponente soll ein Fortschritt in der Entwicklung von benutzerzentrierten, technisch effizienten Webanwendungen sein. Als Reaktion auf die Einschränkungen bestehender Lösungen entsteht diese Selektionskomponente und erweitert das Kolibri Toolkit um eine wichtige und flexible Funktionalität.
Die meisten anderen Konkurrenz-Auswahlkomponenten benötigen möglicherweise eine längere Einarbeitungszeit, da eine Dokumentation durchgelesen werden muss oder die Anwendung sehr komplex sein kann. Weiter haben viele Library-Lösungen Abhängigkeiten zu weiteren Komponenten oder externen Ressourcen wie einem Package Manager. Die Verwendung von Bibliotheken hat zur Folge, dass Vertrauen auf die Sicherheit und Verfügbarkeit der Ressourcen erforderlich ist. Anderenfalls ist mit Abstrichen zu rechnen. Einige Konkurrenzprodukte erfordern viele Codezeilen, um die gewünschte Funktionalität zu erreichen. Häufig ist die Definition spezifischer Konfigurationen notwendig. Diese Kenntnisse unterstützen die Recherche nach Umsetzungsmöglichkeiten.

3 Weg zur Auswahlkomponente

Auf dem Weg der Auswahlkomponente kommen diverse Methoden zum Einsatz. Der Abschnitt 3.1 beschreibt die theoretischen Errungenschaften. Die Kapitel 3.2 bis 3.4 zeigen die Anwendung der recherchierten Methoden, sowie die Entscheidungsprozesse im Detail. Hierbei bezieht sich das Prototyping auf die Gestaltungsvorgänge der Komponente. Auf den wichtigen Zwischenschritt des repetitiven Testing folgen die implementativen Vorgehensweisen.

3.1 Recherche

Dieses Kapitel bietet einen Einblick in die verwendeten Methoden, welche in der technischen Recherche auf dem Weg zur Auswahlkomponente Verwendung finden. Es beschreibt die im Konzept verwendeten LoFi- und HiFi-Prototypen, sowie das Refactoring, Dokumentieren des Codes sowie die angewendeten Patterns.

3.1.1 Prototyping

Ein Hilfsmittel, Ideen und Konzepte zu visualisieren, ist das Prototyping. Dies setzt sich aus Prototyp und Testing zusammen, welche gleichwertige Anwendung in dieser Methode finden. Die erarbeiteten Ansätze durch Nutzertests zu analysieren, ist ein wichtiger Bestandteil. Prototypen unterstützen die Entwicklung, um Anforderungen bildlich darzustellen. Alternative Entwürfe vermeiden eine frühzeitige Festlegung auf eine spezifische Idee. Usability-Tests bieten die Möglichkeit verschiedene Gestaltungsformen vor der Implementation zu prüfen.
Der Prozess ist in Iterationen mit folgenden Schritten gegliedert:

- Fokus
- Szenario
- Konzipieren und Erstellen
- Evaluieren

Die erste Phase dient zur Klarstellung, welches Ziel mit dem Prototyp erreicht werden soll. Zudem klärt sich der Umfang und die Darstellungstreue des Durchgangs. Im nächsten Schritt formt sich die Benutzergruppe des zu gestaltenden Bausteins. Dadurch bestimmen sich die Hintergrundkenntnisse des Users. Das Konzipieren und Erstellen beinhaltet die Entwicklung des bzw. der Prototypen und Lösungsvarianten im aktuell definierten Rahmen. Um die Iteration abzuschliessen, gilt es, eine Evaluation vorzubereiten und durchzuführen.

Über die Zeit entstehen mehrere Versionen der Komponente, welche helfen, die Anforderungen zu spezifizieren. Jeder Prototyp zeigt eigene Vor- und Nachteile, welche es zu analysieren und priorisieren gilt. Das Durchspielen der Varianten zeigt auf, an welchen Stellen weitere Anpassungen nötig sind. Hierbei unterscheiden sich Lo-Fi- und Hi-Fi-Prototypen, welche in den folgenden Unterkapiteln weiter beschrieben sind.

3.1.1.1 Lo-Fi Prototyp

Die Entwicklung einer UI-Komponente startet gewöhnlich mit diversen Lo-Fi-Prototypen. Lo-Fi steht für Low Fidelity und beinhaltet im Allgemeinen das Interaktions- und das Informationsdesign. Mit der Architektur der Informationen entsteht der Grundstein, wobei Inhalte einfach zugänglich sein sollten. Die Visualisierungen können von Skizzen bis Wireframes gehen.

Methoden von Lo-Fi-Prototypen sind Papierprototypen, Mockups oder Story Boards. Erstere gelten als schnellste Methode und sind grobe Skizzen des UI, um die Usability zu prüfen. Der Fokus liegt auf dem Konzept und ist durch minimalste Technologie möglich. Es eignet sich, Abläufe, Funktionalität, Layout und Inhalte festzulegen. Wireframes dienen zur schematischen Darstellung des UI und sind reine Schwarz-Weiss Skizzen. Als Grundlage für ein Usability Walkthrough zeigt diese Methode nur die Konzeption der Benutzeroberfläche. Ein häufig verwendetes Tool ist Balsamiq. Werden die Skizzen in eine logische Abfolge gesetzt, entsteht ein Story Board. Der Ablauf entsteht durch Benutzerfeedbacks oder in Zusammenarbeit mit den Usern. Hierbei zeigen sich die Interaktionsschnittstellen. Simulationen helfen bei der Klärung von Anforderungen und zeigen sehr früh Usability-Probleme auf.

(Figma, 2024) Vorteile der Low Fidelity Prototypen sind das schnelle Sammeln von Benutzerfeedbacks und das Reduzieren des Entwicklungsrisiko. Noch während des laufenden Betriebs fliessen Änderungen und Korrekturen ein. Mit der Zeit sind die Mockups weitestgehend ausgearbeitet. Im nächsten Entwicklungsschritt entstehen Hi-Fi-Prototypen.

3.1.1.2 Hi-Fi Prototyp

Dieser Schritt im Designprozess startet mit ein bis zwei Resultaten der Lo-Fi Prototypen und entwickelt daraus diverse Hi-Fi-Varianten. Hi-Fi steht für High Fidelity und kümmert sich um die Details des Designs. Der Level reicht von detailreichen Skizzen bis voll interaktiven Prototypen. Diese Modelle können auf Papier, in HTML/CSS/JS oder in einem Tool wie Figma umgesetzt sein. Sie zeigen anhand statischer Screens die realen Oberflächen und Interaktionen auf.

Der Fokus kann auf dem Visual Design oder der Interaktion liegen. Wenn das Visuelle eine grössere Rolle spielt, ist der Detailgrad sehr nahe an der künftigen Umsetzung. Im anderen Fall liegt die Interaktion im Mittelpunkt und der visuelle Detailgrad kann niedrig oder hoch sein.

Um das reale Erscheinungsbild zu simulieren, sind bereits Inhalte des Endprodukts im Einsatz. User-Tests unterstützen hier bei der Entwicklung des UI-Designs und der Abstimmung des Benutzerflusses. Dieser Schritt vermeidet nicht nur kostspielige Produktfehler in der Umsetzung.

Der Hi-Fi-Prototyp arbeitet bestenfalls mit einem Designsystem. (Interaction Design Foundation, 2024) Darin sind Standards definiert, welche in der aktuellen Gestaltung immer wieder auftreten. Die Konventionen enthalten wiederverwendbare Komponenten und bieten eine Konsistenz über das Design. Die diversen Richtlinien dienen zur Vereinheitlichung der Prototypen.

Ein möglicher Standard kann der Weissraum sein. Hierbei ist der Unterschied zwischen dem Makro-Leerraum für die Wertigkeit und dem Mikro-Weissraum für die Lesbarkeit zu bedenken. Wird nur eine einzelne Web-Komponente entwickelt, spielt hauptsächlich die Mikro-Ebene eine Rolle. Die Abstände geben dem Baustein eine Struktur. Ein Raster hilft bei der Definition der Abstände, als auch der Grösse von einzelnen Objekten. Weiter definiert das System die Typografie, welche die Schriftart, -grösse, -schnitt und -familie beinhaltet. Proportionen sind weitere Elemente eines Designs, welche standardisiert sein können. In den richtigen Verhältnissen geben diese dem Nutzer ein angenehmes Gefühl.
Farben spielen bei der Gestaltung eine zentrale Rolle. Sie ergeben in der richtigen Kombination folgende Harmonien (Abbildung 3.1):

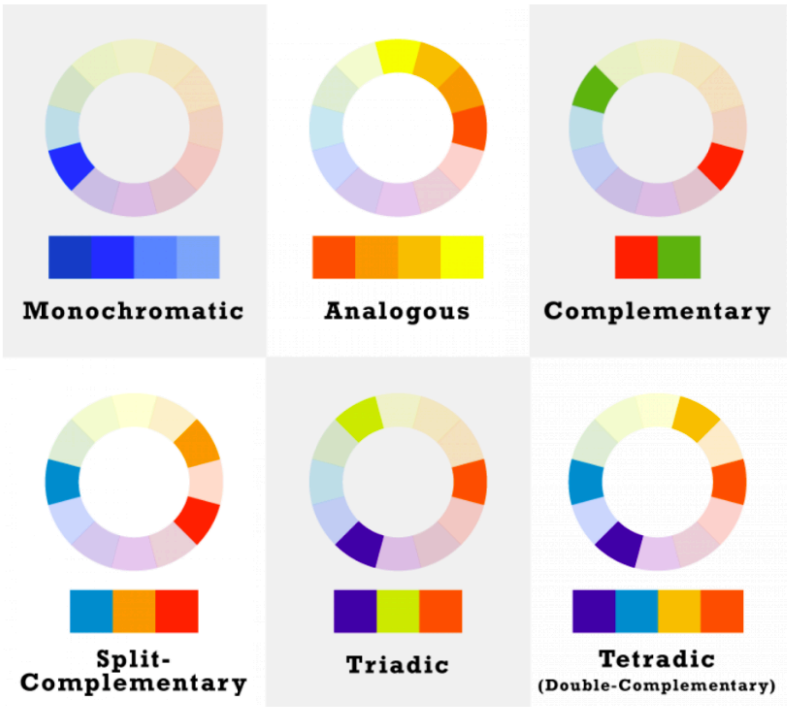
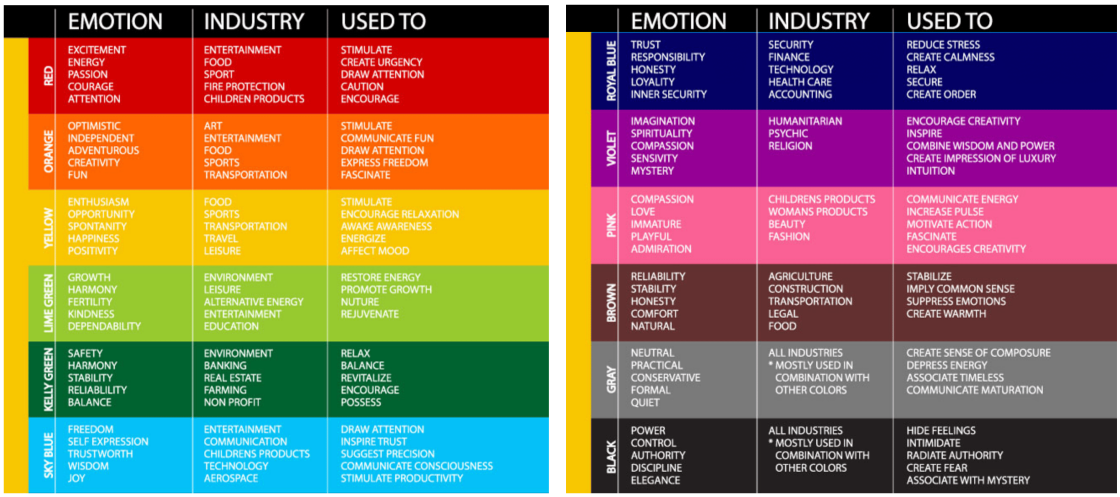


Abbildung 3.1 - Farb-Harmonien (Appendix G)

Die Kolorierung kann in verschiedenen Tönungen und Sättigungen auftreten. Zudem gilt es, die Farbpsychologie (Abbildung 3.2) zu beachten, damit der Nutzer das gewünschte Gefühl übermittelt erhält.



	EMOTION	INDUSTRY	USED TO
RED	EXCITEMENT ENERGY PASSION COURAGE ATTENTION	ENTERTAINMENT FOOD SPORT FIRE PROTECTION CHILDREN PRODUCTS	STIMULATE CREATE URGENCY DRAW ATTENTION CAUTION ENCOURAGE
ORANGE	OPTIMISTIC INDEPENDENT ADVENTUROUS CREATIVITY FUN	ART ENTERTAINMENT FOOD SPORTS TRANSPORTATION	STIMULATE COMMUNICATE FUN DRAW ATTENTION EXPRESS FREEDOM FASCINATE
YELLOW	ENTHUSIASM OPPORTUNITY SPONTANITY HAPPINESS POSITIVITY	FOOD SPORTS TRANSPORTATION TRAVEL LEISURE	STIMULATE ENCOURAGE RELAXATION AWAKE AWARENESS ENERGIZE AFFECT MOOD
LIME GREEN	GROWTH HARMONY FERTILITY KINDNESS DEPENDABILITY	ENVIRONMENT LEISURE ALTERNATIVE ENERGY ENTERTAINMENT EDUCATION	RESTORE ENERGY PROMOTE GROWTH NUTURE REJUVENATE
KELLY GREEN	SAFETY HARMONY STABILITY RELIABILITY BALANCE	ENVIRONMENT BANKING REAL ESTATE FARMING NON PROFIT	RELAX BALANCE REVITALIZE ENCOURAGE POSSESS
SKY BLUE	FREEDOM SELF EXPRESSION TRUSTWORTH WISDOM JOY	ENTERTAINMENT COMMUNICATION CHILDRENS PRODUCTS TECHNOLOGY AEROSPACE	DRAW ATTENTION INSPIRE TRUST SUGGEST PRECISION COMMUNICATE CONSCIOUSNESS STIMULATE PRODUCTIVITY

	EMOTION	INDUSTRY	USED TO
ROYAL BLUE	TRUST RESPONSIBILITY HONESTY LOYALITY INNER SECURITY	SECURITY FINANCE TECHNOLOGY HEALTH CARE ACCOUNTING	REDUCE STRESS CREATE CALMNESS RELAX SECURE CREATE ORDER
VIOLET	IMAGINATION SPIRITUALITY COMPASSION SENSIVITY MYSTERY	HUMANITARIAN PSYCHIC RELIGION	ENCOURAGE CREATIVITY INSPIRE COMBINE WISDOM AND POWER CREATE IMPRESSION OF LUXURY INTUITION
PINK	COMPASSION LOVE IMMATURE PLAYFUL ADMIRATION	CHILDRENS PRODUCTS WOMANS PRODUCTS BEAUTY FASHION	COMMUNICATE ENERGY INCREASE PULSE MOTIVATE ACTION FASCINATE ENCOURAGES CREATIVITY
BROWN	RELIABILITY STABILITY HONESTY COMFORT NATURAL	AGRICULTURE CONSTRUCTION TRANSPORTATION LEGAL FOOD	STABILIZE IMPLY COMMON SENSE SUPPRESS EMOTIONS CREATE WARMTH
GRAY	NEUTRAL PRACTICAL CONSERVATIVE FORMAL QUIET	ALL INDUSTRIES * MOSTLY USED IN COMBINATION WITH OTHER COLORS	CREATE SENSE OF COMPOSURE DEPRESS ENERGY ASSOCIATE TIMELESS COMMUNICATE MATURATION
BLACK	POWER CONTROL AUTHORITY DISCIPLINE ELEGANCE	ALL INDUSTRIES * MOSTLY USED IN COMBINATION WITH OTHER COLORS	HIDE FEELINGS INTIMIDATE RADIATE AUTHORITY CREATE FEAR ASSOCIATE WITH MYSTERY

Abbildung 3.2 - Tabelle der Farb-Psychologien (Appendix G)

Sind die Farben gewählt, folgt die Prüfung, dass die Accessibility gegeben ist. Diese gliedert sich in die Hauptgebiete Seheinschränkung, Höreinschränkung und Einschränkungen im Text- und Informationsverständnis. Für Sehbehinderte hat die Farbe eine wichtige Bedeutung.
Um die Icons einheitlich zu halten, sollten diese im System als Library angeboten werden. Darin ist zu achten, dass alle dem selben Schema folgen – sei es stroke oder filled.

Mikro-Interaktionen unterstützen den Nutzer bei einer Aktion. Der Trigger löst eine Aktion aus, welche vom Benutzer oder System stammen. Sobald die Aktion startet, liegen Regeln vor, was passieren soll. Ein visuelles, auditives oder haptisches Feedback teilt dem Benutzer mit, was geschehen ist. Im letzten Schritt ist definiert, welche Auswirkungen eine Änderung der Bedingungen zur Folge hat. Diese Designaspekte beeinflussen den User – in manchen Situationen sogar unbewusst.

Zeitpunkte Micro Interactions einfliessen zu lassen sind:
- Daten-Eingabe
- Anzeige einer kommenden Aktion
- Aktueller Systemstatus
- Animierte Buttons

3.1.2 Refactoring

(Gillis, 2024) Refactoring zielt darauf ab, den Code übersichtlicher und lesbarer zu gestalten. Eine Schritt für Schritt Umstrukturierung ermöglicht es, komplexe Passagen ohne Funktionalitätsänderung, die Verständlichkeit zu erhöhen. Diese Methode hilft Fehlerquellen zu finden, welche zuvor in der Komplexität versteckt waren. Teilweise werden gewisse Schwachstellen sogar gelöst. Zweck ist es Abhängigkeiten aufzulösen und bestenfalls unabhängige Komponenten zu erstellen. Die bessere Lesbarkeit unterstützt eine spätere Wartung und erhöht die Widerstandsfähigkeit. Der Prozess ist iterativ. Am Ende ist der Code idealerweise möglichst komprimiert und besteht aus extrahierten Komponenten, welche klar in ihrem Verwendungszweck sind. Die Verschachtelungen sind grösstenteils aufgelöst oder umgeschrieben. Wichtig ist, nach einem Refactoring alle Funktionen erneut zu testen.

3.1.3 JSDoc

Dieser Teil geht nur auf die wichtigsten Eckpunkte von JSDoc ein. Es beschreibt nur die verwendete Komponente und nicht das volle Potenzial. Die folgenden Passagen referenzieren das Thema JSDoc von rstacruz (2024).

Um Funktionen zu erklären und den Anwender zu unterstützen, kann folgendes Code Snippet 3.1 mit angepasstem Inhalt über der Code-Komponente platziert werden.

Code Snippet 3.1

```
/**
 * This function writes the string of the parameter defaultValue
repete
 * times in an {@link Array}.
 *
 * @param { !String } defaultValue - mandatory string to define the
insert value
 * @param { Number } repeat        - optional number to define the
number of
 *                                        entries in the array,
default 1
 * @return { Array<String> } a list of repeat number of entries with
the value
 * @throws { NumberOutOfRangeException } - happens if repeat is
negative
 *
 * @example
 *       const myList = createArray("foo", 5);
 */
const createArray = (defaultValue, repeat = 1) => { doWork };
```

Zeile 2 des Code Snippet 3.1 gibt eine zusammenfassende Beschreibung, welchen Zweck die Funktion hat. Optional kann in einem weiteren Absatz mehr Details zu der Funktion gegeben werden. Das @link referenziert bereits existierende Typen, um in den IDEs einfacher navigieren zu können. Die Zeilen 4 und 5 listen die Parameter @param und deren Zweck auf. Zugleich legen sie den erlaubten Typ der Argumente in den geschweiften Klammern fest. Zeile 6 definiert den Rückgabetyp @return, welcher im Erfolgsfall erstellt wird. Fehler, die von der Funktion geworfen werden (Zeile 7), erhalten eine Beschreibung startend mit dem @throws. Als weitere Unterstützung stellt @example (Zeile 9) ein Beispiel bereit.

Code Snippet 3.2

```
/**
 * This function creates a default person with the name from arg val.
 *
 * @private
 * @constructor
 * @template _T_
 * @param { _T_ } val - value to save in the name property
 * @return { Person } a default person
 */
const Person = (val) => { name: val };
```

Wenn die Funktion ein neues Objekt wie in Code Snippet 3.2 erstellt, erhält diese die Markierung @constructor (Zeile 5). Um generische Funktionen zu dokumentieren, wird @template (Zeile 6) zu Hilfe genommen und als Typ in der weiteren Beschreibung verwendet. Die Sichtbarkeit der Funktion oder Variable beschränkt sich mit @private (Zeile 4) auf das aktuelle File.

JSDoc erweitert JavaScript mit der Möglichkeit Variablen und Konstanten Typen zuzuweisen oder gar eigene Typen zu erstellen.

Code Snippet 3.3

```
/**
 * @typedef { "January" | "February" | "March" | "April" } Month
 */

/**
 * @typedef { object } Date
 * @property { Number } day
 * @property { Month } month
 * @property { Number } year
 */

 /**
 * @type { Date }
 */
let currentMonth = { day: 8, month: "March", year: 2020 };
```

Zeile 2 des Code Snippet 3.3 definiert durch @typdef einen neuen, einfachen Typ, welcher die Monate Januar bis April als Werte zulässt. Es ist sogar möglich verschiedene Typen wie Zahlen und Texte zu kombinieren. Der neue Typ mit @typedef auf Zeile 6 definiert ein Objekt, welches aus einem Tag, einem Monat und einem Jahr besteht. Hierbei stammt der Monat (Zeile 8) aus dem zuvor erstellten Typ von Zeile 2 und die anderen Felder mit @property identifizieren sich als Zahlen (Zeilen 7 und 9). Die eigenen Typen können wie die vordefinierten verwendet werden (Zeile 13).

3.1.4 Projector Pattern

(König, 2024) Das Projector Pattern (Abbildung 3.3) ist eine Erweiterung des MVC Pattern. Zu Model, View und Control kommen Projectors, welche mit den Controllern verschiedene Views generieren.

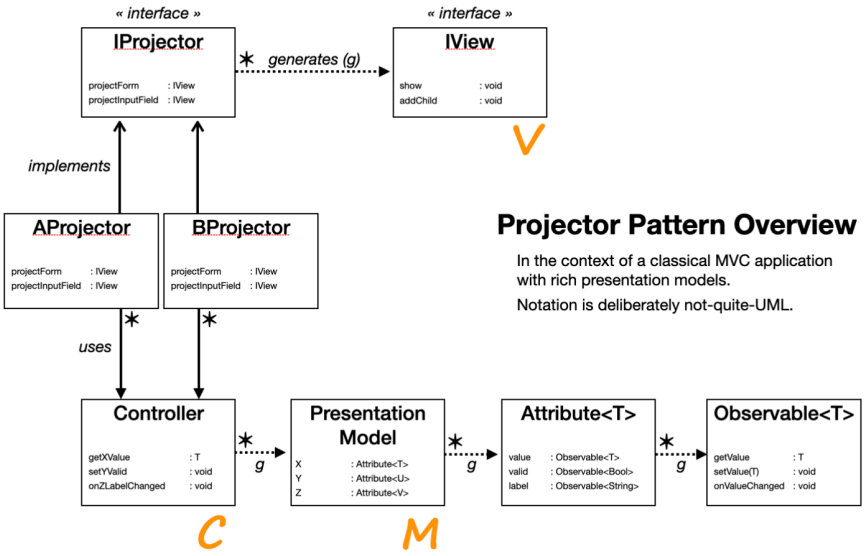


Abbildung 3.3 - Schema der Projector Pattern (König, 2024)

Die Projectors binden durch die Controller das Model bzw. PresentationModel an die View. Die Art der Bindung hängt von den Eigenschaften der View und dem Zweck des Projectors ab. Controller können mehrere Presentationsmodelle generieren, behalten diese aber privat. Dafür veröffentlichen sie Methoden, die von den Projectors verwendet werden. Da die Zugriffe auf Modelle alle über die Controller laufen, setzt dieser alle Geschäftsregeln durch. Ein PresentationModel generiert Attribute. Diese wiederum generieren benötigte Observables, welche mehr Informationen als nur den Wert enthalten. Wichtig ist, dass der View keine Kenntnis von den anderen Komponenten erlangt.

3.1.5 Master-Detail View

Eine Master-Detail View kommt zur Anwendung, wenn eine Liste von Elementen angezeigt wird. Zugleich ist ein spezifisches Element davon mit allen Informationen in der View sichtbar. Der Master View (auch Explorer genannt) enthält die Liste. In dieser sind nur wenige Informationen oder nur ein Wert pro Eintrag ersichtlich.

3.2 Prototyping

Die folgenden Kapitel beschreiben das Vorgehen in der Prototyping-Phase. Angefangen von der Suche eines geeigneten Beispiels über diverse Lo-Fi bis hin zu den Hi-Fi Prototypen entwickeln sich mehrere Strukturen und Varianten. Mit dem Voranschreiten der Prototypen zeichnen sich immer mehr Details aus und beantwortete Fragen fliessen ein.

3.2.1 Beispiel finden

Eine Möglichkeit, die Aufgabenstellung zu visualisieren, ist Anwendungsbeispiele für die Auswahl-Komponente zu finden. Die verschiedenen Ideen helfen die Problematik des Standard-Dropdown aufzuzeigen. Sobald die Fragestellung geklärt ist, folgt das Analysieren aller Beispiele. Die Fokussierung auf ein gemeinsames Beispiel erleichtert das weitere Vorgehen. Die Arbeit mit konkreten Daten ist einfacher. Dadurch sind die Prototypen bzw. Varianten nicht nur analysierbar sonder auch vergleichbar.

Die Liste "alle Länder der Welt" bietet sich als passendes Beispiel an. Dieser Datensatz enthält eine grosse Datenmenge von ca. 250 Werten. Das konkrete Beispiel findet sich auf vielen Formularen wieder. Zudem deckt es die in Kapitel 1.1 erwähnte Problematik ab.

3.2.2 Lo-Fi Prototypen

Um schnell eine Vielzahl von Ideen zu entwickeln und zu visualisieren, ist das aktive Verwenden des Lo-Fi Prototyping die beste Möglichkeit. Dadurch entstehen zwischen 7 und 10 Konzepte mit jeweils 2 bis 5 Varianten. Wegen den Feinheiten des Designs nicht vom Weg abzukommen, ist das oberste Ziel.

Die Konzentration liegt auf der grundlegenden Anordnung und den erforderlichen Elementen der Komponente. Effizient klären sich hierbei entscheidende Fragen zur Funktionalität und zum Nutzererlebnis. Die unbedingt benötigten Elemente als auch deren Anordnung innerhalb der Benutzeroberfläche stehen im Fokus. Ideen schnell zu Papier gebracht (Abbildung 3.4) helfen verschiedene Lösungsansätze unmittelbar zu visualisieren und zu bewerten.

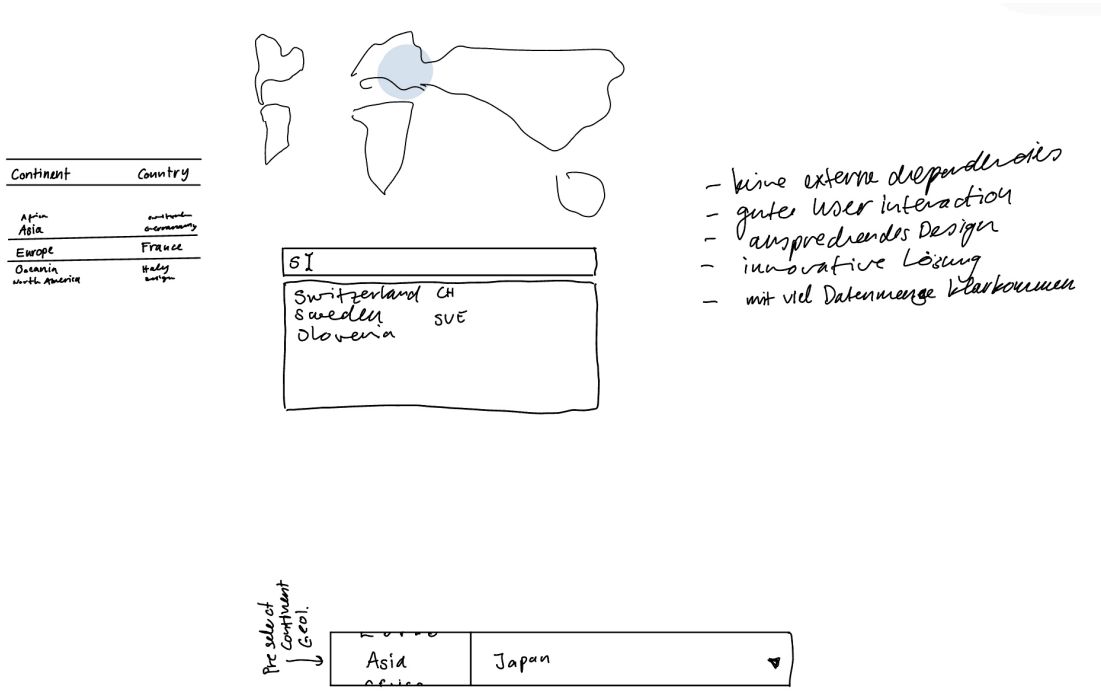


Abbildung 3.4 - Brainstorming Skizzen

Die Verwendung von Lo-Fi Prototyping erweist sich als besonders vorteilhaft für die Förderung der Kreativität. Diese Methode erleichtert das frühzeitige Einholen von Feedback. Dadurch minimiert sich der Zeitaufwand und die Kosten für Änderungen.
Als Teil der Lo-Fi Prototypen entstehen HTML-Skizzen (Abbildung 3.5 und 3.6). Hierbei liegt – wie auf Papier – die Anordnung der Elemente im Mittelpunkt. Schnell fällt auf, welcher Prototype eher unübersichtlich wirkt (Abbildung 3.6) und welcher viel Platz einnimmt (Abbildung 3.5). Die Code-Ordnung ist nicht relevant.

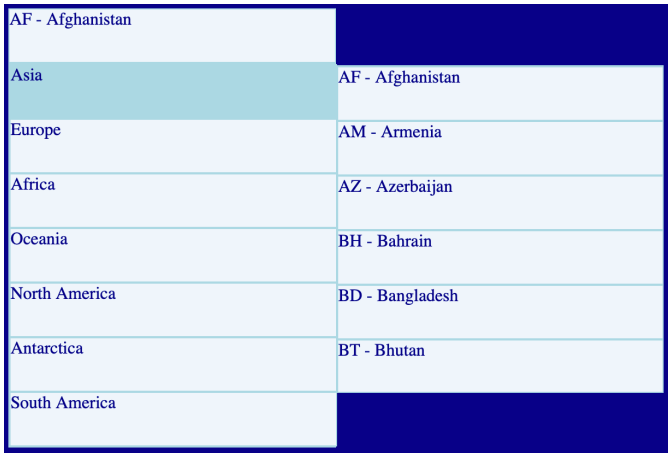


Abbildung 3.5 - HTML LoFi-Prototyp 1

Abbildung 3.6 - HTML LoFi-Prototyp 2

3.2.2.1 Papierprototyp

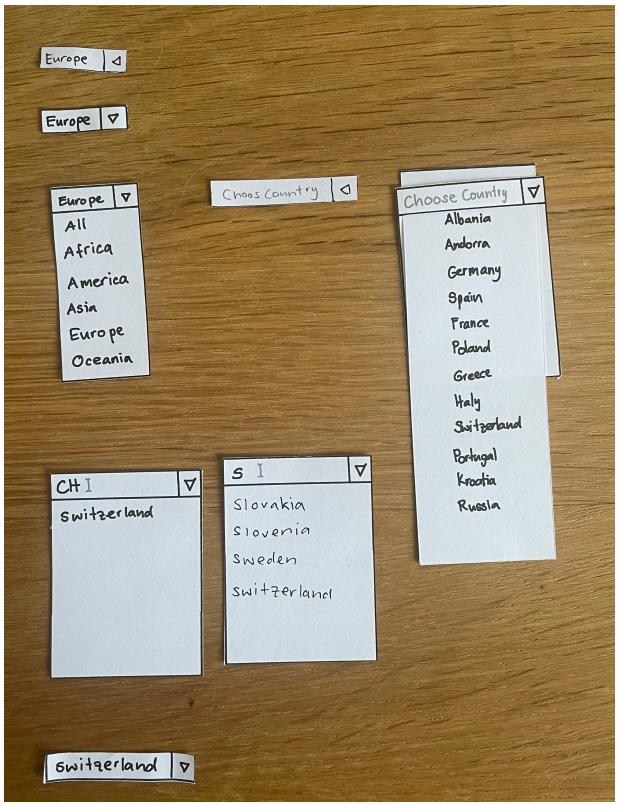


Abbildung 3.7 - Papierprototyp

Um die Interaktionen und das Benutzererlebnis der Entwürfe zu verfeinern, kommen im UX/UI-Design-Prozess Papierprototypen zum Einsatz. Durch die Erstellung von drei Lo-Fi Prototypen aus Papier gelingt die detaillierte Darstellung der geplanten Interaktionen. Dies hilft die Bedienung Schritt für Schritt nachzuvollziehen.

Dieser Ansatz ermöglicht, die Handlungen der User präzise zu durchdenken und darzustellen. Erst wenn die Abläufe geklärt sind, ist der Übergang zur digitalen Umsetzung sinnvoll. Durch die Dokumentation der Interaktionen auf Papier zeigen sich verschiedene Benutzerpfade. Hierbei ermöglichen sich Simulationen als auch das Ausprobieren der Benutzeroberfläche. Dies begünstigt frühzeitig mögliche Probleme und Herausforderungen in der Benutzerführung zu erkennen und anzugehen.

In User-Tests erweist sich das Papier-Prototyping als besonders wertvoll. Die direkten Benutzerfeedbacks sind eine einfache und kostengünstige Möglichkeit Korrekturen früh einzubringen. Zugleich testet dieser Schritt die grundlegenden Designkonzepte. Das analoge Durchlaufen der Papiermodelle bietet einen grossen Gewinn an Erkenntnissen über die Benutzererfahrung. Zusätzlich stellt der Vorgang sicher, dass Designentscheidungen auf den Bedürfnissen und Erwartungen der künftigen Benutzer basiert.

Insgesamt ist das Papier-Prototyping ein entscheidender Schritt im Designprozess. Dabei unterstützt der analoge Lo-Fi Prototyp die Entwicklung einer nutzerzentrierten und intuitiven Benutzeroberfläche. Daraus entsteht eine solide Grundlage für den Hi-Fi Designansatz.

3.2.3 Hi-Fi Prototyp

Bei der Übernahme des Papierprototypen ins Figma entstehen zwei Varianten. Von Anfang an liegt der Fokus darauf in Komponenten zu arbeiten. Dadurch lassen sich Anpassungen einfach ergänzen und umsetzen. Das im Figma bereits enthaltene Designsystem hilft bei der Entwicklung der Hi-Fi Prototypen. Das monochrome Grau bzw. Violett stellt sich im ersten Farbkonzept in den Mittelpunkt. Ziel der Grautöne ist es, dezent und elegant zu wirken. Das Violett greift die Primärfarbe des Kolibri auf und arbeitet mit dessen Schattierungen.
Die erste Variante bildet grösstenteils den Papier-Prototypen strukturell ab. Die violetten Stylings stellen die Status der Elemente dar. Der geschlossene Zustand der Auswahlkomponente erscheint in einem einfachen Aufbau (Abbildung 3.8). Dabei zeigt sie den Platzhalter oder das selektierte Land mit einem rechtsbündigen "X" an, als auch einen Pfeil für das Öffnen der Komponente.

Abbildung 3.8 - Geschlossenes Dropdown Variante 1, Farbkonzept 1

Die Ausarbeitung des Dropdown-Inhalts gestaltet sich aufwendiger. Die möglichen Zustände stellen sich bei der Analyse der Komponente heraus. Beim Designen teilen sich die Status in normal, selektiert, fokussiert und selektiert-fokussiert auf. Für eine Unterscheidung dessen erhält jeder ein eigenes Aussehen (Abbildung 3.9).

Abbildung 3.9 - Status Elemente [normal, selektiert, fokussiert, selektiert-fokussiert] Variante 1, Farbkonzept 1

Die Selektion muss auf den ersten Blick gut sichtbar sein, daher erscheint eine dezente Hintergrundfarbe logisch. Denn diese sticht, ohne aufdringlich zu wirken, gut heraus. Der Fokus erhält ein dezenteres Erscheinungsbild, muss aber zugleich mit der Selektion kombinierbar sein. Daraus resultiert die Entscheidung das Element durch einen Rahmen umzugestalten. Da dieser nicht so viel Fläche abdeckt und einen Kontrast zur Selektion bieten soll, ist die Violett-Schattierung kräftiger.

Die geöffnete Auswahlkomponente setzt sich aus vorbereiteten Elementen zusammen (Abbildung 3.10). Die Kopfzeile enthält dabei nur noch oben einen Border-Radius, um die Listenbox zu ergänzen. Dadurch zeigen sich die beiden Teil-Komponenten als Einheit. Um dem Benutzer den Zustand der Listenbox anzuzeigen, dreht sich der Pfeil in der Kopfzeile.

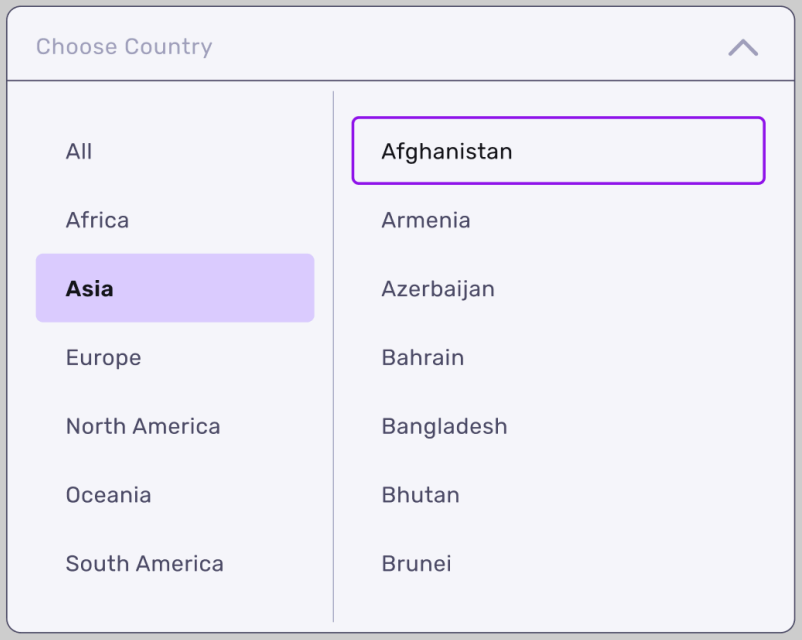


Abbildung 3.10 - Offenes Dropdown Variante 1, Farbkonzept 1

Parallel zu der ersten Version (Abbildung 3.10) entfaltet sich eine zweite kompaktere Variante (Abbildung 3.11). In dieser ist dieselbe Kopfzeile ersichtlich, jedoch mit anders gerichteten Pfeilen. Im geschlossenen Dropdown zeigt der Pfeil nach links und im offenen nach unten. Um die Auswahlkomponente (Abbildung 3.11) kompakter darstellen zu können, sind die Kontinente als Kürzel in Lesezeichen orientierten Reitern angeordnet. Der aktive Kontinent erhält einen dunkleren Hintergrund, um den selektierten Zustand anzuzeigen.

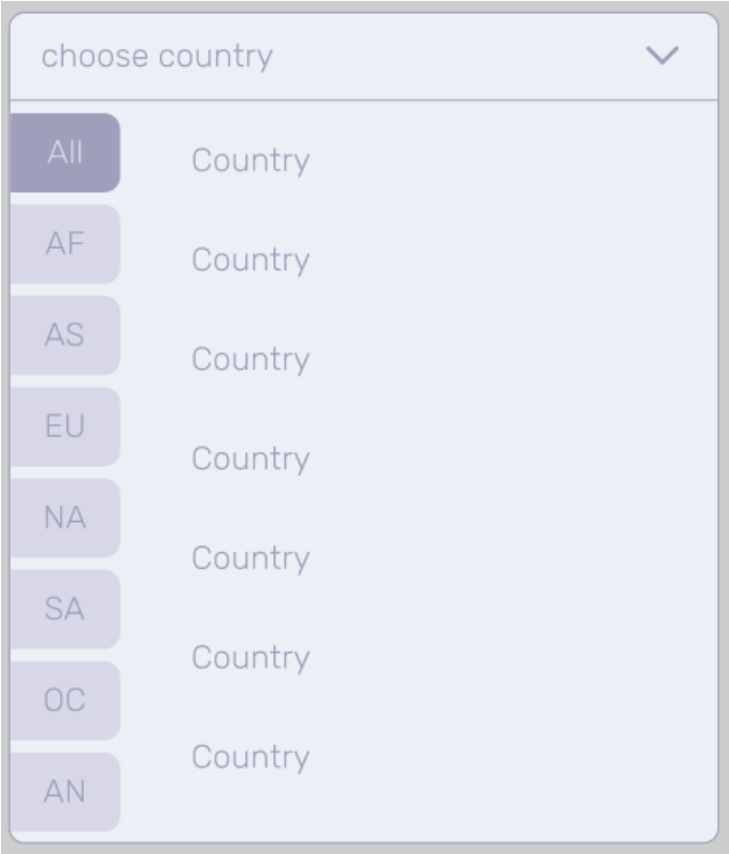


Abbildung 3.11 - Offenes Dropdown Variante 2, Farbkonzept 1

Beim Vergleich der beiden Varianten zeigt sich der erste Hi-Fi Prototyp als geeigneter. Bei der späteren Generalisierung findet sich nicht für jede Kategorie eine Abkürzung. Die erste Variante der unterschiedlichen Pfeile ist den meisten Usern geläufiger.
Eine intensivere Auseinandersetzung mit der Bedienbarkeit zeigt auf, dass die Accessibility der Komponente eine Überarbeitung benötigt. Der Rahmen für den Fokus und die Ähnlichkeit der Schattierungen sind in dieser Hinsicht eher ungeeignet (Abbildung 3.9). Für Personen mit einer Sehbeeinträchtigung bietet das gewählte monochrome Farbkonzept einen zu schwachen Kontrast. Die Namen der Status geben den Hinweis auf die neuen Farben. Im Kolibri existieren bereits die Selektionsfarbe Gelb und die Akzentfarbe Rosa, welche einen guten Kontrast bieten. Zusätzlich wird Gelb in der Farbpsychologie (Abbildung 3.2) für eine energetische Stimmung genutzt. Diese beiden Farben bilden das zweite Farbkonzept. Für eine bessere Erkennung des Fokus-Elements löst sich der Rahmen zu Gunsten der Schriftfarbe auf. Daraus entstehen die Elemente der Abbildung 3.12. Diese wiederum sind in der Abbildung 3.13 im Einsatz zu sehen.

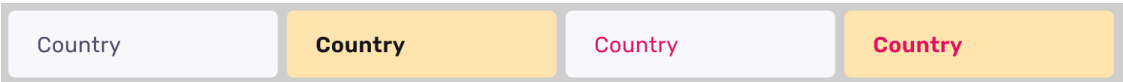


Abbildung 3.12 - Status Elemente [normal, selektiert, fokussiert, selektiert-fokussiert] Variante 1, Farbkonzept 2

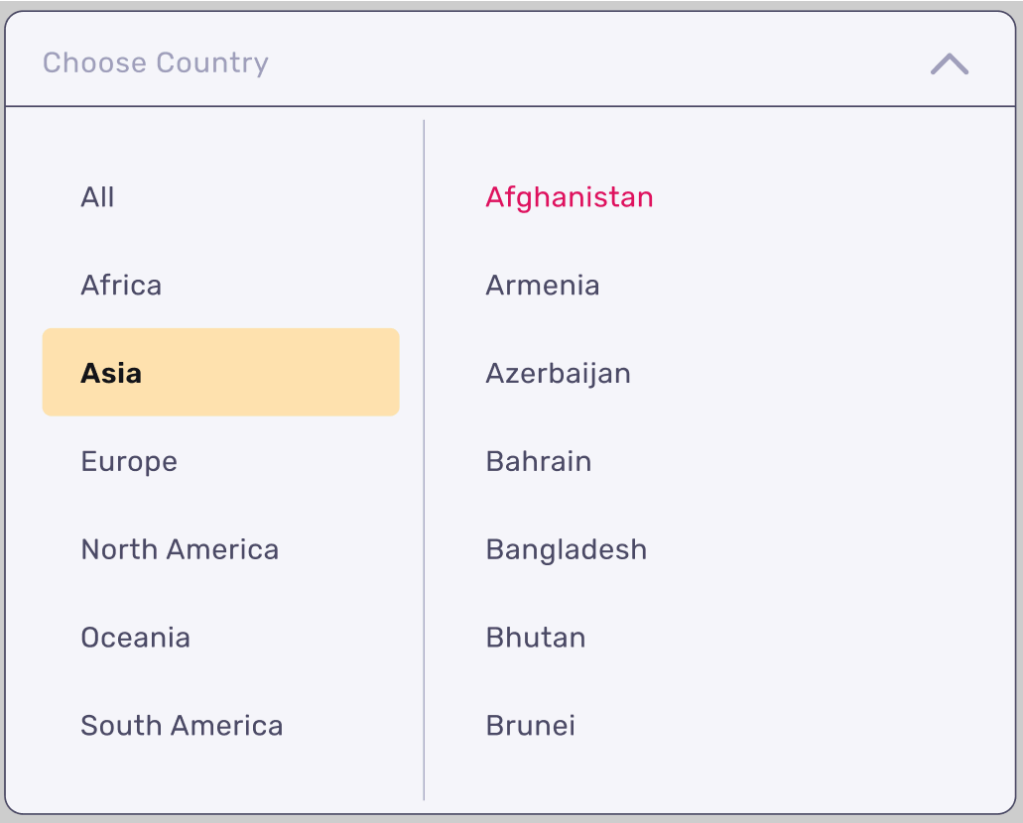


Abbildung 3.13 - Offenes Dropdown Variante 1, Farbkonzept 2

3.2.4 Interaktionen

In der Lo-Fi und Hi-Fi Prototyping Phasen sammeln sich verschiedene Interaktionsmöglichkeiten und -abläufe. Dabei spielt der Hintergrund der befragten Nutzer eine tragende Rolle.

Ein typischer Alltagsnutzer verwendet in der Interaktion grösstenteils die Maus. Diese ermöglicht unter anderem die Aktionen Klick, Hover und Scrollen. Der Klick auf ein Element – Kontinent oder Land – setzt den Status dessen auf selektiert. Dadurch erhält jener Bestandteil das dazu passende Design. Das zuvor in der selben Spalte selektierte Element wechselt den Status auf normal. Bei dem Klick auf ein Land wird dessen Wert in der Kopfzeile eingesetzt. Wenn sich die Maus über einem Kontinenten oder Land bewegt oder befindet, steht dieses mit dem dafür definierten Stil im Fokus. In der ganzen Listenbox besitzt nur ein Eintrag den Status fokussiert. Ein Klick auf den Pfeil auf der rechten Seite der Kopfzeile öffnet bzw. schliesst das Dropdown, je nach Zustand vor der Aktion. Bei einem Klick auf das "X" neben dem ausgefüllten Wert leert sich das Feld in der Kopfzeile. Zusätzlich hebt sich die Selektion in der Länderspalte auf. Das Scrollen innerhalb der Länderspalte bewegt diese Liste nach oben oder unten.

Auch Tastatur-User kommen bei dieser Komponente auf ihre Kosten. Aktionen mit der Tastatur sind ein weiterer Bestandteil des Interaktion-Designs. Im geschlossenen Zustand des Dropdown bewirkt die Enter-, Leer- oder Pfeiltaste nach unten das Öffnen des Dropdown. Ist die Listenbox einmal sichtbar, dienen die Pfeiltasten zur Navigation zwischen den Elementen. Springt der Fokus auf ein Land ausserhalb des sichtbaren Bereichs, verschiebt sich die Liste um einen Eintrag in dessen Richtung. Enter oder Space selektieren den im Fokus stehenden Wert. Bei einer Länder-Selektion füllt sich die Kopfzeile aus, anderenfalls wechselt der Fokus vom Kontinenten-Eintrag in die Länder-Spalte. Die Escape-Taste schliesst die Listenbox, ohne eine Änderung am eingetragenen Wert vorzunehmen. Backspace und Delete entfernen den Wert aus der Kopfzeile. Tab behält die typische Funktion in Formularen bei und wechselt zum nachfolgenden Input. Zusätzlich schliesst die Aktion die Listenbox. Bei der Verwendung der einzelnen Symbole – wie z.B. Buchstaben – beginnt eine Suche in der Länderliste. In der Liste wird zum ersten mit der Eingabe übereinstimmenden Eintrag gescrollt. Bei der Eingabe der Suche kommt die nachfolgende Strategie zum Einsatz.

Die Debounce-Technik als Schlüsselelement trägt wesentlich zur Effizienz und Benutzerfreundlichkeit bei. Um die Benutzerinteraktionen zu optimieren, bietet es sich an, diese Technologie bei einer umfangreichen Liste strategisch einzusetzen. Die Eingabe hält sich jedoch nur eine kurze Zeit – weniger als eine Sekunde – im Speicher. Nach dem Ablauf dieser Zeitspanne, ab dem ersten Symbol gemessen, fängt die Suche neu an. Genauere Implementationsdetails folgen im Kapitel 3.4.2.2.

3.3 Testing

Um Designvarianten zu bewerten, ist die Durchführung von User-Tests unerlässlich. Ein Teil der Nutzerbefragungen basiert auf Papierprototypen. Die Tests zielen darauf ab, die Benutzerfreundlichkeit, Effizienz und intuitive Bedienbarkeit der verschiedenen Prototypen zu ermitteln. Die Evaluation der drei Prototypen – L (Abbildung 3.14), M (Abbildung 3.15) und R (Abbildung 3.16) – findet mit unterschiedlichen Testpersonen statt.

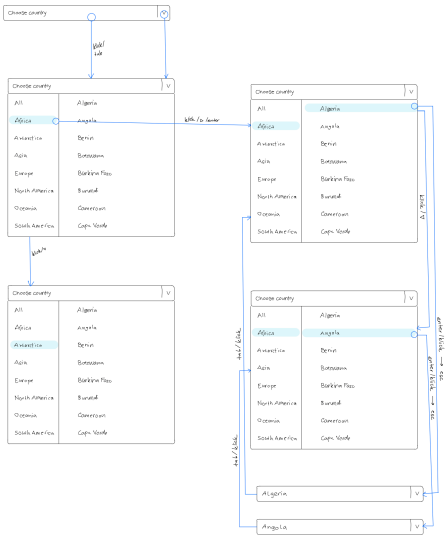


Abbildung 3.14 - Schritt für Schritt Ablauf der Testperson 1

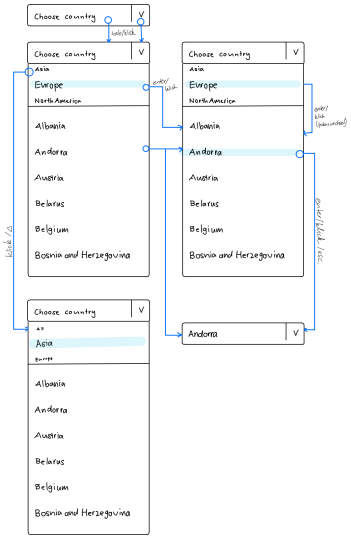


Abbildung 3.15 - Schritt für Schritt Ablauf der Testperson 2

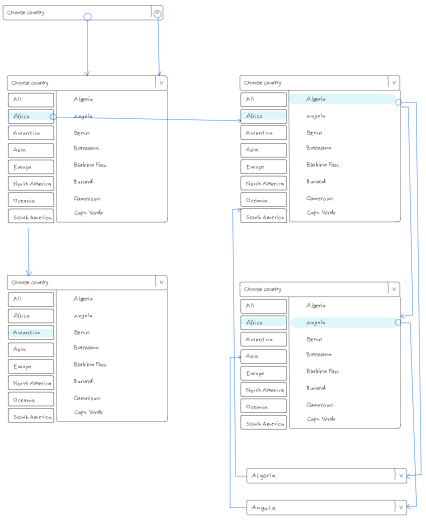


Abbildung 3.16 - Schritt für Schritt Ablauf der Testperson 3

3.3.1 Methodik

Um die Prototypen zu testen, sind die oben erwähnten Testpersonen mit abweichenden Profilen – Robert (junger Alltagsbenutzer, Digital-Native), Sara (semi aufmerksam, nicht schnell auf der Tastatur) und Nick (Alltagsbenutzer/ Student) – ausgewählt. Jeder der drei Nutzer führt eine Reihe von Aktionen durch, wie etwa das Auswählen eines Landes. Daraus resultiert jeweils Feedback mit ihren Erfahrungen.

3.3.2 Ergebnisse und Feedback

Folgende Rückmeldungen unterstützen die weitere Entwicklung der Komponente enorm. Die unterschiedlichen Sichtweisen beheben den eigenen schwarzen Fleck.

3.3.2.1 Testperson 1: Rober

- Prototyp L:
  Als effizienteste Lösung bewertet, da dieser verschiedene Auswahlmöglichkeiten bietet und eine schnelle Navigation ermöglicht.
- Prototyp M:
  Verwirrung über die Notwendigkeit, Kontinente zuerst auszuwählen; unklar, ob Länderkürzel eingegeben werden können.
- Prototyp R:
  Vorauswahl nicht intuitiv; Anwender wollte ursprüngliche Auswahl bestätigen, nicht ändern.

3.3.2.2 Testperson 2: Sara

- Prototyp L:
  Klare und einfache Bedienung, Tastaturbedienung nicht notwendig.
- Prototyp M:
  Einfache Bedienung, aber Tastaturbedienung nicht erforderlich.
- Prototyp R:
  Klar und einfach zu bedienen, keine Tastaturbedienung erforderlich.

3.3.2.3 Testperson 3: Nick

- Prototyp L:
  Klare und direkte Auswahlmöglichkeit, schnelle und intuitive Bedienung.
- Prototyp M:
  Negatives Feedback wegen zu vieler Zwischenschritte und aufgezwungener Auswahl.
- Prototyp R:
  Anfängliche Verwirrung, Vorauswahl sollte klarer sein.

3.3.3 Zusammenfassung des Feedbacks

Das allgemeine Feedback betont die Wichtigkeit einer effizienten und benutzerfreundlichen Gestaltung der Dropdown-Komponente. Die Testpersonen bevorzugen Ansätze, die eine direkte und schnelle Auswahl ermöglichen, ohne unnötige Zwischenschritte oder Verwirrung. Prototyp L erhält insgesamt als auch einzeln die beste Bewertung in der Effizienz und Benutzerfreundlichkeit.

3.3.4 Schlussfolgerungen

Die Ergebnisse zeigen, dass eine optimale Dropdown-Komponente eine Balance zwischen Effizienz, Einfachheit und intuitiver Bedienung bieten muss. Die Möglichkeit, direkt und ohne Umwege ein Land auszuwählen, löst ein positives Gefühl aus. Dies unterstreicht die Bedeutung einer gut durchdachten Benutzeroberfläche, die die Bedürfnisse der Endnutzer in den Vordergrund stellt. Das Finale Design muss diese Erkenntnisse berücksichtigen, um eine optimale Benutzererfahrung zu gewährleisten.

3.3.5 Testing mit den Kunden

Die Kunden testen die gleichen Papierprototypen wie die Testpersonen aus Kapitel 3.3.1. Sie bewerten die Benutzerinteraktion und die technische Umsetzung, um spezifisches Feedback zur Weiterentwicklung der Komponente zu geben. Erfahrungen und Fachkenntnisse sind in die Bewertung mit eingeflossen.

3.3.5.1 Feedback und Anpassungen

- Autovervollständigung:
  Die Dropdown-Komponente soll nicht sofort bei Fokussierung öffnen, sondern erst bei der Eingabe von Buchstaben. Indem das spätere Öffnen die Übersichtlichkeit erhöht und gleichzeitig die Auswahl zu treffen beschleunigt, könnte sich die Benutzererfahrung verbessern.

- Tastaturinteraktion:
  Die Pfeiltasten zur Navigation in der Dropdown-Liste nutzen. Beispielsweise könnte die Pfeil-nach-unten-Taste die Liste öffnen, um eine flüssigere Bedienung zu ermöglichen.

- Browser-Autocomplete-Standard:
  Standards wie die Browser-Autocomplete-Funktion können genutzt werden, um die Bedienung intuitiver zu gestalten.

- Leertaste als Öffnungsmechanismus:
  Die Integration der Leertaste als zusätzliche Option zum Öffnen der Komponente, um die Bedienbarkeit zu erleichtern.

- Fokus- und Schliessverhalten: Den Fokus innerhalb der Komponente auch bei Mouse-Out zu behalten, aber bei einem Klick ausserhalb zu schliessen, um eine konsistente Benutzererfahrung zu gewährleisten.

- Backspace für Korrekturen:
  Die Möglichkeit, mit Backspace eine Auswahl rückgängig zu machen.

- Weitere technische Empfehlungen:
  Wo möglich const verwenden und die Implementierung des vorhandenen DebounceInput für die Sucheingaben nutzen.

3.4 Implementation

Dieses Kapitel behandelt das Vorgehen während des Implementations-Prozesses. In der Prototyping-Phase entsteht der erste HTML Prototyp. Durch das Anwenden des Refactoring löst sich dieser chaotische Code auf. Im iterativen Prozess entstehen sechs aufgeräumte und dokumentierte Files.

3.4.1 Prototyping in HTML

Parallel zur Ausarbeitung der Details des Figma Prototypen entsteht ein Lo-Fi HTML-Prototyp (Appendix D). Ziel des Codes ist eine erste, rudimentäre Implementation zu erstellen. Damit zeigt sich ein erster Beweis für die Umsetzbarkeit der konzeptionell geplanten Komponente.

Code Snippet 3.4

```
<html>
    <head>
        <!-- head tags -->
        <style>
            /* all styles here */
        </style>
    </head>
    <body>
        <!-- html view -->
        <script src="countries.js"></script>
        <script defer>
            /* all js here */
        </script>
    </body>
</html>
```

Dieser Prototyp (Code Snippet 3.4) bestehend aus einer Datei enthält den gesamten HTML, CSS (Zeile 5) und JavaScript (Zeile 12) Code. Wegen der grossen Datenmenge (Zeile 10) resultiert die Entscheidung diesen Code als einzigen in eine externe Datei (Zeile 10, Appendix C) auszulagern. Dies vereinfacht die Verwendung in weiteren Prototypen. Ein Auszug aus der Datenmenge von Appendix C ist in Code Snippet 3.5 ersichtlich.

Code Snippet 3.5

```
const countryList = [
    {
        country: "Afghanistan",
        code: "AF",
        code2: "AFG",
        domain: "af",
        continent: "Asia",
    },
    {
        country: "Albania",
        code: "AL",
        code2: "ALB",
        domain: "al",
        continent: "Europe",
    },
    {
        country: "Algeria",
        code: "DZ",
        code2: "DZA",
        domain: "dz",
        continent: "Africa",
    },
    ...
];
```

Wenn sich das Konzept im Lo-Fi Prototyp als gut erweist, entwickelt sich der Ansatz zu einem Hi-Fi Prototyp (Appendix E) weiter. Dabei kommt der erste tiefere Kontakt mit der Codebasis zustande.

Code Snippet 3.6

```
@import url("../../docs/css/kolibri-base.css");
@import url("../../docs/css/kolibri-light-colors.css");
@import url("../../docs/css/kolibri-light-fonts.css");

:root {
    --color-selected: var(--kolibri-color-select, hsl(46, 90%, 84%));
    --color-selected-secondary: var(--kolibri-color-shadow, hsl(46,
80%, 92%));
    --color-focused: hsl(322, 73%, 52%);
    /* more style variables here */
}
/* more styles here */
```

Als wichtige CSS Dateien erweisen sich kolibri-base.css, kolibri-light-colors.css und kolibri-light-fonts.css des Kolibri-Repository (Appendix B) . Diese finden sich in den Importen im <style>-Tag wieder (Code Snippet 3.6).

Bei den JS-Dateien stechen die Dateien aus dem Ordner simpleForm heraus. Die drei Files des SimpleInput zeigen den Aufbau einer vorhandenen Formularkomponente. Diese Struktur dient beim späteren Refactoring (Kapitel 3.4.2) als Vorlage bzw. Beispiel.

Der Hi-Fi Prototyp von Kapitel 3.2.3 beinhaltet eine Bedienung mit der Tastatur (Code Snippet 3.7) und bietet eine Möglichkeit zur Suche.

Code Snippet 3.7

```
const inputElement = /* create element */;
inputElement.onkeydown = (e) => {
    switch (e.code ?? e.key ?? e.keyCode) {
        case "ArrowLeft": /* handle action */
        /* more navigation actions */
        case "Enter": /* handle action */
        /* more actions */
        default: /* nothing happens */
    }
};
```

Die Navigation (Code Snippet 3.7) mit den Pfeiltasten, sowie weiteren typischen Aktionstasten ist über ein Key Event geregelt. Diese Umsetzung kümmert sich nur im die grundlegenden Tasten, welche von den meisten anderen Elementen ebenfalls unterstützt werden.

3.4.2 Refactoring inkl. Entwicklung

In der weiteren Entwicklung des HTML Hi-Fi Prototypen aus dem Appendix E kommt das Refactoring in iterativen Schritten zum Zuge. Bei der ersten Überarbeitung liegt der Fokus auf dem Aufteilen des gesamten Codes in mehrere Dateien. Die Zerlegung schafft eine bessere Struktur. Es entsteht pro Technologie ein File. In der weiteren Entwicklung geschieht das unabhängige, repetitive Refactoring der drei Dateien.

Das CSS teilt sich in zwei Dateien (CSS-Dateien im Appendix F). Eine Datei dient der Verwaltung der Variablen, welche für die Komponente zuständig sind. Das Design der einzelnen Selektoren steht im zweiten File. Da dieser Teil noch sehr viel Code enthält, hilft eine weitere Iteration der Überarbeitung. Diese prüft die einzelnen Properties auf ihre Notwendigkeit und erstellt eine Ordnung bzw. Gruppierung der Selektoren. In Kombination zum Refactoring wird das Design optimiert und entdeckte Fehler korrigiert.

Code Snippet 3.8

```
<!-- link to js code -->
<script src="./dropdown.js" type="module"></script>
```

Code Snippet 3.9

```
import { countryList } from "./countries.js";

/* JS code */
```

Auf der anderen Seite vereinfacht die Modularisierung (Code Snippet 3.8 & 3.9) des erstellten JS-Files die weitere Überarbeitung des Codes. Der zweite Refactoring-Durchlauf löst die Komplexität aus dem Prototypen ein wenig auf. Dies geschieht indem die Variablen eine saubere Benennung erhalten. Zudem entstehen Funktionen mit einem klaren Ziel, welche an den ursprünglichen Stellen aufgerufen werden. Im gleichen Schritt entwickelt sich der erste Prototyp der Suche. Während dieses Durchgangs entsteht parallel der erste Teil des JSDoc. Die Umsetzung des Projector Pattern als auch der Debounce Suche folgt im Zusammenhang mit weiteren Iterationen des Refactoring.

3.4.2.1 Projector Pattern

Das Projector Pattern teilt den komplexen JS-Code in drei modulare Komponenten (JS-Dateien im Appendix F). Um das UI testbar zu implementieren, gesellt sich zu diesen Files noch ein weiteres mit dem JS-Starter Code dazu. Die klare Trennung der Verantwortlichkeiten unterstützt die Wartung als auch die Weiterentwicklung des Codes. Die Wiederverwendbarkeit wird dadurch gesteigert. Da diese Struktur bereits beim SimpleInput umgesetzt ist, dient diese Komponente als gute Vorlage. Die Abhängigkeiten zwischen den einzelnen Dateien minimiert sich. Bei der Umsetzung des Patterns reduziert sich der Code im HTML-File auf wenige Zeilen (Code Snippet 3.10).

Code Snippet 3.10

```
<!-- main container for dropdown -->
<div class="countrySelectionView dropdown" id="container">
    <!-- dynamic generated by JavaScript code -->
</div>
```

Die Analyse des SimpleInput offenbart, dass bereits ein Debounce-Input existiert. In der folgenden Passage ist beschrieben, wie diese zum Einsatz kommt.

3.4.2.2 Debounced Searching

Im Refactoring- und Optimierungsprozess der Suche bietet sich eine Debounce-Funktion an. Diese kommt ohne ein sichtbares UI-Feld aus. Der default case des Key Events (Code Snippet 3.7) für die Auswahlkomponente definiert die Aktion von Tasteneingaben einzelner Symbole wie z.B. Buchstaben. Das Debounce fügt alle Eingaben innerhalb einer Zeitspanne zu einem Suchbegriff zusammen. Dieser wiederum kommt bei der Suche zum Zuge. Die vorhandene Debounce-Komponente bietet sich an, um den Suchbegriff zu verwalten (Appendix F, Datei choiceInputProjector.js). Zusätzlich löst die Eingabe einen Trigger für die Suche in der Länder-Liste aus. Das dafür notwendige Input-Feld dient rein zur Verwaltung der Daten und zum Triggern der Suche. Das UI hat keine Kenntnisse von diesem Input und ist dadurch vor Manipulationen geschützt.

Bei einer weiteren Analysierung der soweit entstandenen Komponente zeigt sich ein Master-Detail Aufbau. Dieses Muster lässt sich in einer nächsten Iteration des Refactoring umsetzen.

3.4.2.3 Master-Detail View

Jede dieser JS-Dateien erhält bei der Umsetzung des Master-Detail Pattern Änderungen. Eine sinnvolle Reihenfolge für die Überarbeitungen der Pattern-Komponenten startet beim Model (Code Snippet 3.11), gefolgt vom Controller (Code Snippet 3.12) und zum Schluss der Projector (Code Snippet 3.13).

Code Snippet 3.11

```
export { ChoiceDetailModel, ChoiceMasterModel };

const ChoiceDetailModel = ({value, placeholder, label, name}) => {
    /* detail model logic */
};
const ChoiceMasterModel = ({elementList, sectionElement, focusObject})
=> {
    /* master model logic */
};
```

Code Snippet 3.12

```
import { ChoiceDetailModel, ChoiceMasterModel } from "./
choiceInputModel.js";

export { ChoiceDetailController, ChoiceMasterController };

const ChoiceDetailController = (args) => /* detail controller logic */
const ChoiceMasterController = (categoryColumn, elementColumn, timeout
= 800) =>
                                  (args) => /* master controller logic
*/
```

Code Snippet 3.13

```
const projectChoiceInput = (detailController, masterController,

formCssClassName) => {
    /* projector logic */
};
```

Die ersten zwei Files trennen ihren Inhalt in einen Master und einen Detail (Code Snippet 3.11 & 3.12) Teil auf. Dadurch entstehen jeweils zwei getrennte Models und zwei unabhängige Controller. Der Projector (Code Snippet 3.13) erwartet durch angepasste Parameter nun je einen Master- und einen Detail-Controller. Im Starter Code ist es nun notwendig die Models als auch die Controller separat zu erstellen und gemeinsam dem Projector zu übergeben.

Der Detail View erhält die Daten der Kopfzeile der Komponente. Die Informationen sind sowohl im geschlossenen als auch im offenen Zustand sichtbar. Die Master Ansicht ist nur in letzterem Status im UI sichtbar. Zugleich verwaltet es den Zustand als auch den Debounce-Text der Komponente. Folgende detaillierte Aufteilung entwickelt sich während des Refactoring und der Umsetzung des Patterns.

Detail:
- Selektiertes Element aus Werte-Liste (Value)
- Platzhalter
- Label
- Name (für Formulare)

Master:
- Liste aller Elemente
- Selektionsobjekt aus Kategorie und Value
- Fokusobjekt aus aktiver Spalte und Element (Kategorie oder Value)

3.4.3 Fazit

Das Prototyping im HTML zeigt schnelle Erfolge, ohne sich sehr lange mit den Details aufzuhalten. Der schrittweise Ausbau unterstützt die Entwicklung. Mit dessen wiederholten Beschäftigung wird der Code immer klarer. Anfangs komplex erscheinende Stellen der Codebasis oder eigener Dateien, erscheinen verständlicher. Die erhaltene Klarheit des Codes hilft bei der Dokumentation. Das Refactoring unterstützt die Entwicklung insofern, dass sich Fehler durch das Umschreiben offenbaren. Hierbei grenzen sich die Stellen ein und die Bugs lassen sich schneller beheben.

4 Länder-Auswahlkomponente

Die Auswahlkomponente entwickelt sich durch die Verwendung von Lo-Fi-, Papier- und Hi-Fi-Prototypen. Daraus entsteht ein einfach zu bedienendes und schön gestaltetes Dropdown. Dank der verschiedenen Methoden und Nutzertests entsteht ein durchdachtes Design. Im aktuellen Zustand ist die Lösung auf das Beispiel einer Länder-Selektion ausgelegt. Das gut strukturierte und farblich angepasste Erscheinungsbild findet sich im CSS wieder. Zur Vervollständigung kommt die umgesetzte Ausgestaltung der Interaktion, sowie die Logik hinter der Komponente dazu.

4.1 Design

Das komplette Design basiert auf dem Beispiel einer Länder-Auswahl-Komponente.
Im groben Aufbau besteht das neue Dropdown aus zwei Teilen – die Kopfzeile und die Content-Box. Die Kopfzeile besteht aus einem Feld für das selektierte Land und einem Button für das Öffnen und Schliessen der Komponente (Abbildung 4.1). Ist ein Wert enthalten, lässt sich dieser mit dem angezeigten "X" löschen (Abbildung 4.2). Wird das Dropdown geöffnet, erscheint eine zweigeteilte Box. Diese zeigt die Kontinente auf der linken und die Länder des jeweilig selektierten Kontinents auf der rechten Seite an (Abbildung 4.3). Die Content-Box ist ein Popover und überdeckt nachfolgende Webseiten-Elemente. Die einzelnen Kategorien bzw. Länder sind jeweils untereinander angeordnet. Ein allfälliger Überfluss, wie bei der Menge aller Länder, versteckt sich hinter der Scrollfunktion. Dadurch nimmt die Komponente keine Übermasse in der Grösse an.

Abbildung 4.1 - Geschlossenes Dropdown ohne Selektion

Abbildung 4.2 - Geschlossenes Dropdown mit Selektion Switzerland

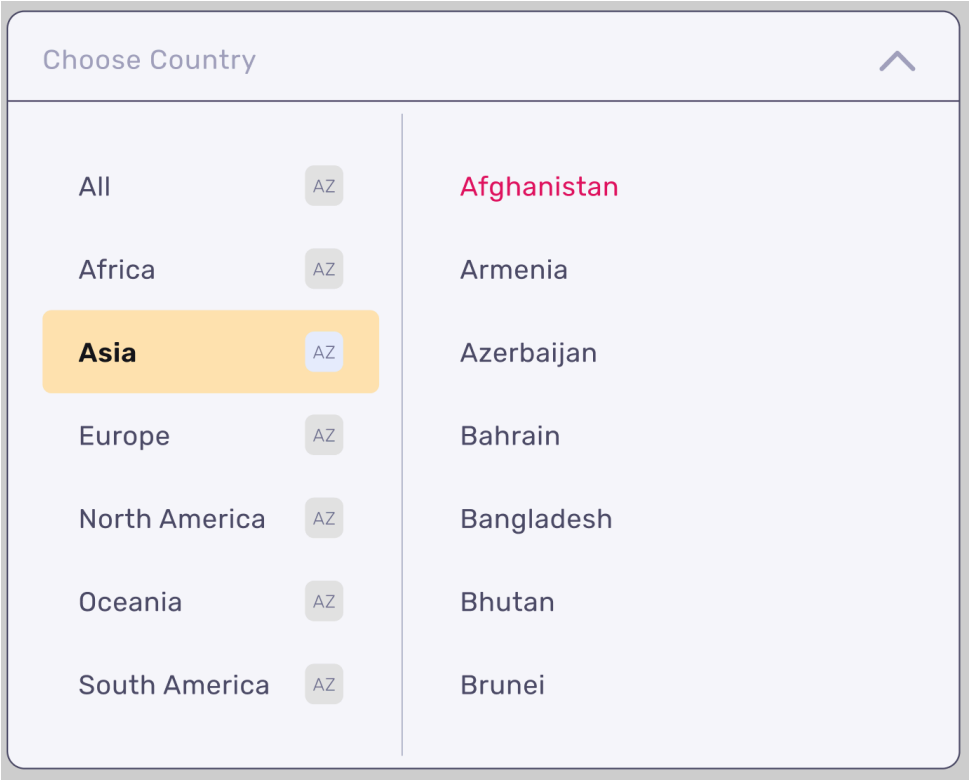


Abbildung 4.3 - Offenes Dropdown mit Selektion Asia & Fokus Afghanistan

Über die diversen Varianten in der Entwicklungszeit zeigen sich die benötigten Status. Die vier festgelegten Zustände sind normal, selektiert, fokussiert und selektiert-fokussiert. Zur Unterscheidung erhalten diese jeweils ein eigenes Aussehen (Abbildung 4.4). Die Benennungen der Status ist auf das Designsystem abgestimmt. Das normale Element soll nicht aufdringlich wirken und sich gut in den Webseiten-Verlauf einbinden. Die aktuelle Selektion hingegen ist prominenter hervorgehoben. Somit erkennt der Nutzer seine momentane Wahl. Als Selektionsvariable existiert in der vorhandenen Farbgestaltung des Kolibri ein mittel-dunkles Gelb. Schwache Farben, wie die gewählte Gelbschattierung, bieten als Schrift- oder Rahmenfarbe einen schlechten Kontrast. Deswegen ist das Einfärben des Elementhintergrunds die beste Wahl. Steht ein Baustein im Fokus, erhält dieser die Akzentfarbe der Kolibri, welche als Pink-Schattierung erscheint. Da der Farbton kräftig ist, eignet er sich gut als Schriftfarbe und lässt sich für den Zustand selektiert-fokussiert (Abbildung 4.4) kombiniert anwenden.

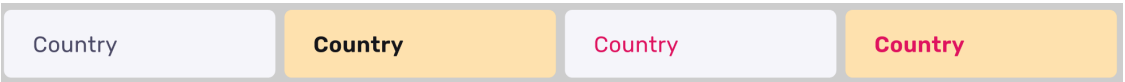


Abbildung 4.4 - Status der Länder [normal, selektiert, fokussiert, selektiert-fokussiert]

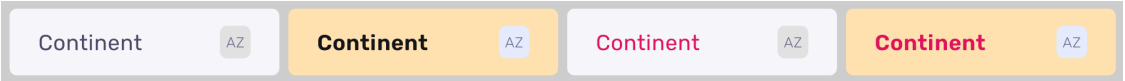


Abbildung 4.5 - Status der Kontinente [normal, selektiert, fokussiert, selektiert-fokussiert]

Die einzelnen Kontinente in der Liste bieten dem Anwender jeweils eine Zusatzinformation (Abbildung 4.5). Diese steht in der rechtsbündigen Informationsbox und zeigt die Anzahl der Länder der jeweiligen Gruppe. Um ein überfülltes Erscheinungsbild zu vermeiden ist diese zusätzliche Angabe (Abbildung 4.5) bei den Ländern (Abbildung 4.4) nicht vorhanden. In der Kopfzeile wird jedoch die Kontinenten-Zugehörigkeit als ergänzende Information angezeigt (Abbildung 4.2).

Durch die User-Tests mit den Papierprototypen zeigt sich, dass Nutzer Webformulare möglichst effizient ausfüllen möchten. Diese Feststellung findet sich im aktuellen Interaktion-Design wieder. Die grosse Menge der Länder im geöffneten Zustand lässt sich mithilfe von Kategorien reduzieren. Die Komponente enttäuscht auch Tastatur-Benutzer nicht. Es bietet sich die Möglichkeit mit der Eingabe der ersten paar Zeichen direkt in die Nähe des gewünschten Landes zu springen. Wenn das Dropdown geschlossen ist, wird der erste gefundene Eintrag über alle Länder direkt als Selektion eingetragen (Abbildung 4.6 - Eingabe “Swi").

Abbildung 4.6 - Bedienung Dropdown via Tastatur

Abbildung 4.6: In geschlossenem Zustand kann die Liste durch Space, Enter oder Pfeil nach unten geöffnet werden. Geöffnet verspricht die Komponente eine angenehme Bedienung, da eine Navigation mit Pfeiltasten unterstützt wird. Die üblichen Bedienelemente – wie Tab für das nächste Eingabefeld, Space bzw. Enter für eine Selektion, ESC zum Schliessen der Content-Box und Backspace bzw. Delete zum Löschen der Selektion – stehen dem Super-User ebenfalls zur Verfügung.

4.2 Implementation

Die Implementierung der Kolibri-Auswahlkomponente kombiniert Intuitivität und benutzerfreundliche UI-Elemente. Sie markiert den Übergang von der Konzeption zur praktischen Anwendung der Ideen und Prinzipien.

Im Bereich CSS nutzt das Projekt bereits vorhandene Kolibri-Dateien und fügt eigene stilistische Erweiterungen hinzu. Dadurch ist die Erfüllung der spezifischen Anforderungen des Dropdown-Elements garantiert. Die Verwendung von vordefinierten Kolibri-Stilen gewährleistet Konsistenz im Gesamtdesign. Währenddessen ermöglichen eigene Erweiterungen eine präzise und massgeschneiderte Nutzererfahrung.

Die JavaScript-Implementierung folgt einem modularen Ansatz. Hierbei sorgen klar definierte Controller, Models und Projectors für eine strukturierte und wartbare Codebasis.
Die Implementierung zeigt sich technisch effizient. Dabei nutzt sie moderne Webtechnologien und optimiert die Performance. Features, wie Debouncing in den Controllern, verbessern die Benutzerfreundlichkeit der Komponente.

4.2.1 CSS

Vordefinierte CSS-Dateien des Kolibri finden im Rahmen dieses Projekts ihren Einsatz. Eigene Stildefinitionen fliessen durch zwei ergänzende Dateien mit ein. Somit ist Konsistenz im Design gewährleistet, während gleichzeitig spezifische Anforderungen des Teilprojekts erfüllt werden.

Kolibiri Basis CSS-Dateien
- kolibri-base.css:
  Diese Datei legt grundlegende Stile wie Schriftarten, Farben, Box-Modell-Regeln und Auswahl-Styles fest. Die Definitionen von Schriftarten wie RobotoSlab und JetBrainsMono sowie Kernfarben und Schatteneffekte sind hier enthalten.

- kolibri-light-colours.css:
  Fokussiert auf Farbdefinitionen, bietet diese Datei eine breite Palette von Farbvarianten in verschiedenen Helligkeitsstufen – unter anderem Lila, Lavendel und Gelb.

- kolibri-light-fonts.css:
  Ergänzt die Basisschriftarten um weitere Stile und das Font-Weight der Rubik-Schriftfamilie, was eine feinere Typografie-Kontrolle ermöglicht.

- testUI.css:
  Speziell für Testzwecke entwickelt, definiert es Stile für Testelemente und -layouts, um eine konsistente Benutzeroberfläche während der Entwicklung zu gewährleisten.

Eigene CSS-Erweiterungen
- kolibri-input-elements-dd.css:
  Konzentriert sich auf die Gestaltung von Dropdown-Elementen. Es definiert Stile für Elemente wie Auswahllisten, Schaltflächen und Eingabefelder. Ein besonders grosser Wert liegt auf Anpassungen wie Abstände, Schriftgrössen und Farben.

- kolibri-input-elements-colours.css:
  Diese Datei erweitert die Farbpalette und passt sie an spezifische Bedürfnisse dieser Komponente an. Es legt Farben für ausgewählte und fokussierte Elemente sowie Schriftarten und Hintergründe fest.

Die vordefinierten Kolibri-CSS-Dateien in Kombination mit den ergänzenden Erweiterungen ermöglichen, eine einheitliche und projektspezifische Benutzeroberfläche zu schaffen. Diese Herangehensweise stellt sicher, dass die Komponente visuell in das Gesamtbild des Kolibri passt. Gleichzeitig erfüllt sie ihre einzigartigen Designanforderungen.

4.2.2 JS

Der JavaScript-Code folgt einem modularen Ansatz, um wiederverwendbare und wartbare Komponenten zu erstellen. Der Schwerpunkt liegt auf der Funktionalität, Klarheit und Einhaltung der Kolibri-Programmierstandards.

- Controller (choiceInputController.js):
  Dienen als Schnittstelle zwischen der Benutzeroberfläche und den Geschäftslogik-Modellen. Der ChoiceDetailController und der ChoiceMasterController verwalten die Interaktionen und Datenflüsse zwischen den UI-Komponenten und den Modellen.

- Models (choiceInputModel.js):
  Repräsentieren die grundlegende Geschäftslogik und Zustände. Das ChoiceDetailModel und das ChoiceMasterModel bieten eine abstrakte Darstellung der Daten und ihrer Beziehungen.

- Projector (choiceInputProjector.js):
  Befasst sich mit der Darstellung der Daten auf der Benutzeroberfläche. Diese Datei enthält Funktionen und Methoden, um die Modelldaten in ein nutzbares Format für die Benutzeroberfläche zu konvertieren und zu präsentieren.

- Starter Code (starter.js):
  Bindet die Komponente und deren Logik in die View ein. Die hier definierten Controller erhalten die Daten des Models. Der Projector empfängt den Master- als auch den Detail-Controller, um die View zu generieren.

- Tests:
  Dienen zur Sicherstellung der Funktionalität aller Komponenten des Projector Pattern als auch der Master-Detail View. Für jeden der vier oben aufgeführten Punkten existiert ein eigenes Test-File. Diese gewährleisten die Zuverlässigkeit und Stabilität des Codes.

Spezifische Implementierungsdetails
- Datenbindung und Zustandsmanagement:
  Um eine reibungslose Benutzererfahrung zu gewährleisten, nutzt die Implementierung reaktive Datenbindungen und das Zustandsmanagement. Änderungen im Modell spiegeln sich unmittelbar in der Benutzeroberfläche wider, und umgekehrt.

- Event Handling:
  Die Event-Handling-Logik ermöglicht eine effiziente und wartbare Interaktion der Benutzeroberfläche.

- Modularität und Wiederverwendbarkeit:
  Der Code ist so strukturiert, dass er leicht in verschiedenen Teilen des Projekts oder in zukünftigen Projekten wiederverwendet werden kann.

Die JavaScript-Implementierung stellt eine effektive Symbiose aus den Standards des Kolibri-Frameworks und den spezifischen Komponenten-Anforderungen dar. Die klare Trennung von Modellen, Controllern und Projectors ermöglicht es, wartbaren und erweiterbaren Code zu schreiben. Dieser bildet die Grundlage für die weitere Entwicklung und Verbesserung dieses Produkts.

4.3 Testing

Die Komponente unterzieht sich Tests, um die Korrektheit und Bedienbarkeit des Ergebnisses zu gewährleisten. Automatisierte Tests konzentrieren sich auf die Implementierung und das Vorhandensein der Elemente in der Ansicht. Benutzertests hingegen decken die Bedienbarkeit und die Erfüllung von Nutzerwünschen auf.

4.3.1 Automatisierte Tests

Zur Sicherstellung der Funktionalität der Komponente existieren Tests für die Bestandteile des Projector Patterns. Abbildung 4.7 zeigt die Resultate der Durchführung der vier Test-Dateien. Die Model-Tests prüfen das Vorhandensein bestimmter Observables. Dies geschieht für beide Modelle, Master und Detail, einmal mit kompletten und einmal mit minimalen Angaben. Bei den Controllern liegt das Augenmerk der Tests auf den Getter und Setter von Attributen, als auch auf den Triggerfunktionen wie onValueChanged. Dabei findet die Überprüfung für den Master- sowie den Detail-Controller ebenfalls getrennt statt. Der Test für den Projector kontrolliert die Werte des Controllers initial und nach dem Ausführen eines Events. Zuletzt wird die View rudimentär geprüft. Hier liegt der Fokus auf der Existenz der Elemente.

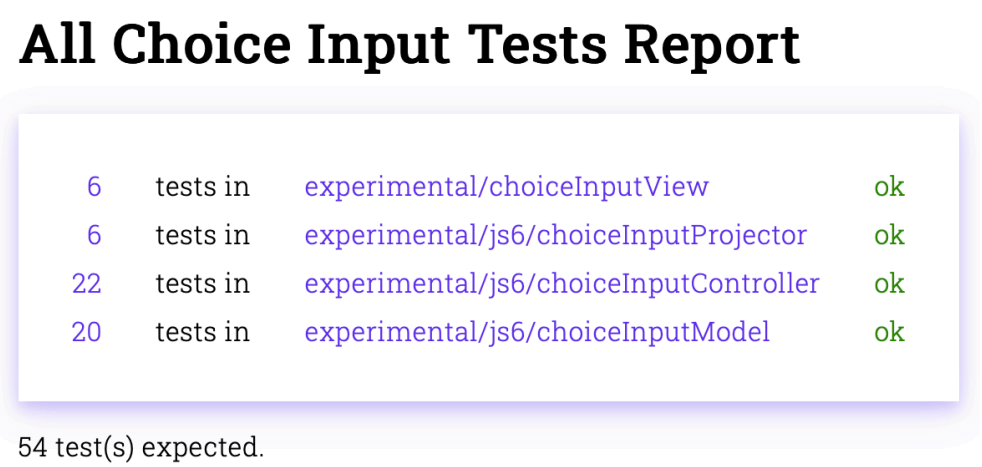


Abbildung 4.7 - Resultate der automatisierten Tests

4.3.2 User Tests

Dieses Unterkapitel dokumentiert das User-Testing der finalen Dropdown-Komponente zur Länderauswahl. Der Schwerpunkt liegt auf der Interaktion der Testpersonen mit dem Prototyp und deren direktem Feedback. Das Ziel ist, die Benutzerfreundlichkeit und Intuitivität der Komponente zu bewerten.

4.3.2.1 Methodik

- Teilnehmer:
  Verschiedene Benutzerprofile, um eine breite Palette an Feedback zu erhalten.

- Aufgabe:
  Ausfüllen eines Formulars mit der Auswahl des Wohnsitzlandes.

- Werkzeuge:
  Beispiel-Formular choiceInputView.html, Maus und Tastatur.

- Messwerte:
  - Benötigte Zeit
  - Aktionen der Testpersonen
  - Reaktion des Prototyps
  - Feedback der Testpersonen

4.3.2.2 Testergebnisse

Ein Beispiel dieses Testdurchlaufs (Tabelle 4.1) zeigt die Interaktion der Testperson Robert mit dem Prototypen der Dropdown-Komponente. Robert, ein junger, alltäglicher Nutzer und Digital Native, verwendet sowohl die Maus als auch die Tastatur für verschiedene Interaktionen. Dabei gibt er wertvolles Feedback zur Benutzerfreundlichkeit und Funktionalität der Komponente. Robert braucht 17.77 Sekunden um das Testformular bzw. die Testaufgabe zu vervollständigen.

Tabelle 4.1 - Ablauf finaler Benutzertest Robert

Tool	Aktion	Reaktion des Prototyps	Feedback
Maus	Klickt in Feld	First name	Feld aktiv
Tastatur	Namen Eintippen	Namen wird in Feld geschrieben	-
Maus	Klick in Feld Name	Feld aktiv	-
Tastatur	Namen eintippen	Namen wird in Feld geschrieben	-
Maus	Klick in Feld Country	Feld aktiv, Dropdown öffnet	-
Tastatur	Switz eintippen	Schweiz erscheint im Feld	Wäre gut, wenn man sehen würde, was man tippt
Maus	Klick bei Schweiz	Komplett ausgefüllt	Zufrieden mit Mausinteraktion

Die folgende Übersicht zeigt die Testergebnisse der Testpersonen (Tabelle 4.2), welche auch bereits die Papierprototypen bewertet haben. Sie geben Einblicke in die Bedienbarkeit und Nutzerfreundlichkeit der finalen Dropdown-Komponente.

Tabelle 4.2 - Testbenutzer finaler Benutzertest

Testperson	Hintergrund	Feedback und Ergebnisse	
Robert	Junger Alltagsbenutzer	Digital-Native	Wünscht visuelles Feedback beim Tippen. Schwierigkeiten mit der Tastaturnavigation beim alleinigen Verwenden der Tastatur. (Hover Maus Problem)
Sara	Semi aufmerksam	nicht so schnell auf der Tastatur	Möchte visuelles Feedback beim Tippen. Ansonsten gut navigierbar mit Maus
Nick	Alltagsbenutzer/ Student	Schwierigkeiten mit Tastaturnavigation; empfiehlt Verbesserungen für den Eingabefeld-Wechsel	=> Feld-Bestätigung/ -wechsel mit Enter gewünscht.

4.3.2.3 Schlussfolgerungen aus den User-Tests

- Durchgängigkeit:
  Es zeigt sich, dass die Konsistenz und Vorhersehbarkeit der Interaktionen für alle Benutzergruppen von großer Bedeutung sind. Insbesondere die visuellen Hinweise und Tastaturinteraktionen müssen konsistent und intuitiv gestaltet sein.

- *Barrierefreiheit*:
  Die Berücksichtigung der Bedürfnisse von Benutzern mit unterschiedlichen Fähigkeiten ist essenziell für die universelle Usability der Komponente.

- *Benutzerführung*:
  Die Ergebnisse weisen darauf hin, dass eine zusätzliche Benutzerführung, wie klare visuelle Hinweise die Bedienung verbessern könnte – insbesondere für weniger erfahrene User. Diesen erscheint es unklar, wo sich sie momentan befindet bzw. was er tippt.

5 Diskussion

Die Implementierung der Dropdown-Komponente demonstriert die Anwendung von Best Practices in der modernen Webentwicklung. Durch die Verwendung modularer, wiederverwendbarer und gut strukturierter Codeelemente ist eine hohe Qualität der Komponente gewährleistet. Dank Benutzerfreundlichkeit und technischer Performance profitieren nicht nur Entwickler sondern auch Endbenutzer. Insgesamt zeigt die Implementierung, dass durch sorgfältige Planung und Umsetzung eine leistungsstarke und gleichzeitig anpassungsfähige Webkomponente entsteht. Diese beeinflusst die Entwicklung von Webanwendungen zum Positiven. Die anfangs definierten Ziele – der Benutzerfreundlichkeit, Anpassungsfähigkeit und technischen Leistungsfähigkeit – sind mit dieser Auswahlkomponente erreicht. Die experimentelle Integration in das Kolibri Toolkit und ihre positive Resonanz in Nutzertests deuten darauf hin, dass die Komponente ein nützliches Werkzeug für zukünftige Webprojekte sein wird. Die Anpassungsfähigkeit und Leistung eröffnen neue Möglichkeiten für die Gestaltung benutzerzentrierter Webanwendungen.

Das Resultat umfasst eine Auswahlkomponente, welche sich auf die Anzeige einer Länderliste mit jeweils dazugehörigem Kontinent spezialisiert. Einige wenige generische Schritte finden bereits ihren Platz. In Zukunft besteht die Möglichkeit die Komponente weiter zu generalisieren, damit der Baustein für beliebige Inhalte zu verwenden ist. Ebenfalls ist in Planung, die Debounce-Suche weiter auszubauen, damit die Kriterien ausgeweitet werden können.

Die Erfahrungen und Erkenntnisse aus diesem Projekt dienen als Grundlage für weitere Innovationen im Bereich der Webentwicklung. Ziel ist, das Kolibri Toolkit kontinuierlich zu verbessern und es zu einem noch leistungsfähigeren Werkzeug für Entwickler und Endnutzer bereitzustellen.

5.1 Future Features

Die Zukunft der Kolibri Dropdown-Auswahlkomponente sieht vielfältig aus. Es sind mehrere Verbesserungen und Erweiterungen geplant, die darauf abzielen, die Funktionalität und Benutzererfahrung weiter zu optimieren. Diese Erkenntnisse basieren aus den bisherigen Tests und dem Feedback der Nutzer.

5.1.1 Weiterführende Tests und Fehlerbehebungen

Ein primärer Fokus liegt auf weiterführenden Tests und der Eliminierung der Bugs. Ein bekanntes Problem ist die gelegentliche Inkonsistenz bei der Tastaturinteraktion, wenn der Mauszeiger über einem Land liegt. Die Komponente soll so angepasst werden, dass die Bestätigung mittels Tastatur immer das visuell hervorgehobene Land auswählt, unabhängig von der Position des Mauszeigers.

5.1.2 Optimierung der Scroll-Logik

Die derzeitige Scroll-Logik in der Länderauswahl erhält eine weitere Optimierung. Ziel ist es, eine intuitivere und flüssigere Navigation durch lange Listen zu ermöglichen. Eine Bewegung der Elemente soll nur geschehen, wenn dies unbedingt nötig ist. Dies ist der Fall, sobald sich das gewünschte Element nicht mehr im sichtbaren Bereich befindet.

5.1.3 Entwicklung weiterer Varianten

Angesichts der Vielseitigkeit der Dropdown-Komponente ist geplant, weitere Varianten zu entwickeln. Diese könnten sich auf andere umfangreiche Datensätze wie Jahreszahlen oder spezifische Kategorien erstrecken. Jede neue Variante muss sich umfangreichen Tests unterziehen, um sicherzustellen, dass sie mindestens die gleiche Benutzerfreundlichkeit und Effizienz wie die aktuelle Länderauswahl bietet.

5.1.4 Abschliessende Bemerkungen

Diese geplanten Entwicklungen zielen darauf ab, die Dropdown-Komponente weiter zu optimieren sowie zu einem flexiblen Werkzeug für Kolibri Toolkit auszubauen. Durch kontinuierliche Verbesserungen und Anpassungen ist die Komponente nicht nur für aktuelle, sondern auch für zukünftige Webentwicklungsprojekte von Wert. Hierfür steht die korrekte Integration der noch experimentellen Komponente in das Kolibri an.

Glossar

Begriff	Definition
Accessibility	Die Praxis, digitale Produkte so zu gestalten und zu entwickeln, dass sie von Menschen mit unterschiedlichsten Fähigkeiten und Behinderungen genutzt werden können.
Auswahlkomponente	Ein UI-Element, das es Benutzern ermöglicht, eine Option aus einer Liste von Möglichkeiten auszuwählen. Typischerweise in Form von Dropdown-Menüs, Radio-Buttons oder Checkboxen.
Balsamiq	Tool für einfaches und schnelles Erstellen von Wireframes. Die Skizzen sind im Lo-Fi Bereich einzuordnen.
Binding	Bindungen definieren Abhängigkeiten zwischen Objekten. Aktionen lösen bei gebundenen Typen Änderungen aus.
Choice-Box	Eine UI-Komponente – ähnlich wie ein Dropdown – welche eine Liste von Optionen zur Auswahl bietet, typischerweise in einem
Debounce	Verzögerte Ausführung einer Funktion. Alle Ereignisse, welche vor dem Ablauf der Verzögerung auftreten, werden verworfen.
Design System	Eine Sammlung wiederverwendbarer Komponenten und Standards, die dazu dienen, die Konsistenz und Effizienz im Designprozess einer digitalen Produktentwicklung zu gewährleisten.
Dropdown	Ein grafisches Steuerelement, das eine Liste von Optionen anzeigt, wenn ein Benutzer darauf klickt. Es wird häufig verwendet, um Platz zu sparen und die Benutzeroberfläche aufgeräumt zu halten.
Figma	Tool für die Erstellung von Hi-Fi Prototypen. Es können einfache Interaktionen aufgezeigt werden. Es bietet die Möglichkeit mit einem Designsystem zu arbeiten.
haptisch	Bezieht sich auf die Wahrnehmung und Interaktion durch den Tastsinn, insbesondere in Bezug auf die Benutzererfahrung mit physischen oder virtuellen Objekten.
Hi-Fi Prototyp	Ein detaillierterer und funktionaler Prototyp, der das tatsächliche Design und die Interaktivität einer Anwendung oder Webseite
hover	Sich mit der Maus auf einem Element befinden bzw. darüber bewegen, ohne eine Taste zu drücken.
Lo-Fi Prototyp	Ein einfacher, oft handgezeichneter Prototyp, der verwendet wird, um Designkonzepte schnell zu visualisieren und grundlegende Funktionen und Layouts zu testen.
Master-Detail View	Dies ist ein Programmieransatz, welcher eine Liste von Werten und Details zu einem der Elemente verwaltet und darstellt.
Mockup	Ein Modell oder eine Nachbildung einer Webseite oder Anwendung, die häufig zur Präsentation des Designs, zur Beurteilung des visuellen Stils und zur Abstimmung mit Stakeholdern verwendet
Observable	Ein Objekt, welches bei Änderung die gebundenen Komponenten informiert und dessen Trigger-Definition ausführt.
Popover	Ein kleines Fenster, das über dem Inhalt der Webseite oder Anwendung erscheint, oft um zusätzliche Informationen oder Auswahlmöglichkeiten anzuzeigen.

Begriff	Definition
Projector Pattern	Ein Programmieransatz zur Erstellung einer mehrschichtigen Benutzeroberfläche. Es wird bei MVC-Applikationen eingesetzt, wobei sich der Projector zwischen View und Controller befindet.
Refactoring	Umschreiben bzw. umformulieren von Code ohne Änderungen der Funktionalität. Die Komplexität löst sich auf.
Selektor	Beschreibt ein HTML-Element, auf welches die darin enthaltenen CSS-Regeln angewendet werden sollen. Bei mehreren Selektoren für das selbe Element spielt die Spezifität eine Rolle.
Trigger	Ein UI-Element oder eine Aktion, die eine bestimmte Funktion oder einen Prozess in einer Anwendung oder Webseite auslöst.
UI	Der Teil einer Software, mit dem der Benutzer interagiert, einschliesslich Bildschirmlayouts, Übergänge, Schnittstellenanimationen und jedes einzelne visuelle Element.
Usability Walkthrough	Aufzeigen von Handlungsabläufen. Es wird die Erlernbarkeit von der Bedienung des Produkts ermittelt. Es deckt Differenzen zwischen der Sicht des Benutzers und des Entwicklers auf.
UX	Die Erfahrung, die ein Benutzer beim Interagieren mit einer Anwendung oder Webseite hat, einschliesslich Benutzerfreundlichkeit, Zugänglichkeit und Effizienz.
Wireframe	Ein einfaches Layout, das die grundlegende Struktur und Komponentenplatzierung einer Webseite oder App darstellt, oft verwendet in den frühen Phasen des Designs.

Quellenverzeichnis

Figma (2024, Januar 01). Low-fidelity prototyping: What is it and how can it help?. Resource library. https://www.figma.com/resource-library/low-fidelity-prototyping/
Figma (2024, Januar 01). What is high-fidelity prototyping—and how can it help?. Resource library. https://www.figma.com/resource-library/high-fidelity-prototyping/
Gillis, A. (2024, Januar 09), Refactoring. Anwendungsentwicklung und -design. https://www.techtarget.com/searchapparchitecture/definition/refactoring
Interaction Design Foundation (2024, Januar 01). Was sind Designsysteme?. Design-Systeme. https://www.interaction-design.org/literature/topics/design-systems
König, D. (2024, Januar 14). Projector Pattern. Efficient Creation of Ambitious User Interfaces. https://dierk.github.io/Home/projectorPattern/ProjectorPattern.html
Maurer Spence, D. (2024, Januar 06). Designing & Selecting Components for UIs. Design. https://uxmag.com/articles/designing-selecting-components-for-uis
rstacruz (2024, Januar 09). Jsdoc cheatsheet. DevHints. https://devhints.io/jsdoc
UX PICKLE (2024, Januar 06). Designing a Data-Heavy UI. https://uxpickle.com/designing-a-data-heavy-ui/
W3Schools (2023, Dezember 01) . HTML <select> Tag. https://www.w3schools.com/tags/tag_select.asp#gsc.tab=0
W3Schools (2023, Dezember 01). HTML <datalist> Tag. https://www.w3schools.com/tags/tag_datalist.asp

Abbildungsverzeichnis

Code Snippet Verzeichnis

Tabellenverzeichnis

Appendix