Dokumentation LEVIKO-XR

Die Architektur der Webseite ist modular aufgebaut, um eine einfache Pflege und Erweiterbarkeit zu gewährleisten. Die Struktur folgt dem Prinzip der Trennung von Verantwortlichkeiten, wobei verschiedene Komponenten und Seiten klar voneinander abgegrenzt sind. Dies fördert Wiederverwendbarkeit und Übersichtlichkeit im Code.

Im Verzeichnis **Components** sind alle wiederverwendbaren Bausteine der Webseite enthalten. Jede Komponente ist in einem eigenen Unterordner organisiert, um spezifische Funktionalitäten zu kapseln und sie leicht zugänglich zu machen. Diese Komponenten reichen von UI-Elementen über Layout-Strukturen bis hin zu speziellen Funktionsbereichen wie Diagrammen und Hooks.

Hier ist eine Übersicht über die wichtigsten Unterordner:

├── src/

│ ├── Pages/

│ ├── Components/

│ │ ├── Accordion/

│ │ ├── Banner/

│ │ ├── customHooks/

│ │ ├── Diagramme/

│ │ ├── Footer/

│ │ ├── Header/

│ │ ├── Hero/

│ │ ├── InfoText/

│ │ ├── Parallax/

│ │ ├── Publikationen/

│ │ ├── stores/

│ │ └── UI/

│ └── assets/

│ ├── images/

* **Accordion**: Beinhaltet alle Module, die sich auf die Implementierung von aufklappbaren Informationsblöcken beziehen.
* **Banner**: Enthält die Logik und das Layout für Banner / Marquees.
* **customHooks**: Eine Sammlung von eigens erstellten React-Hooks, die wiederkehrende Logik abstrahieren und wiederverwendbar machen.
* **Diagramme**: Beinhaltet Komponenten zur Erstellung von Diagrammen und Datenvisualisierungen. -> für die Forschungsseite
* **Footer**: Das Layout und die Inhalte für den Footer der Webseite.
* **Header**: Enthält den Header-Bereich mit Navigationselementen.
* **Hero**: Das Hero-Element, das meist prominent oben auf der Seite angezeigt wird.
* **InfoText**: Komponenten, die zur Anzeige von Informationstexten dienen.
* **Parallax**: Implementierungen von Parallax-Effekten für visuelle Scroll-Interaktionen.
* **Publikationen**: Komponenten zur Darstellung von Publikationen oder Artikeln.
* **stores**: Zentralisierte State-Management-Lösungen mit Zustand <https://zustand-demo.pmnd.rs/> .
* **UI**: Verschiedene Benutzeroberflächenelemente, wie Buttons, Input-Felder, und alle grundlegende UI-Komponenten mit styled-components <https://styled-components.com/> .

Neben den **Components** existieren zwei weitere zentrale Verzeichnisse:

* **Pages**: Dieses Verzeichnis enthält die Seiten der Webseite. Jede Seite repräsentiert eine Route und ist eine Kombination aus den oben beschriebenen Komponenten.
* **Assets**: Alle Images die auf der Webseite verwendet werden.

Im weiteren Verlauf wird auf die einzelnen Seiten eingegangen und wie die Components auf den Seiten integriert sind.

* [Startseite](#Startseite)
* Termine
* [Projekt](#ProjektSeite)
* [Forschung](#Forschungsseite)
* [Publikationen](#PublikationenPage)
* [TeamPage](#TeamPage)

## Startseite

Das sind die Hauptkomponenten, die auf der Startseite verwendet werden. Jede Komponente hat ihre spezifische Funktion und trägt zur Struktur und Gestaltung der Startseite bei.

* [HeroStartseite](#HeroStartseite)
* [MarqueeBanner](#MarqueeBanner)
* [BildText](#BildText)
* [TextBild](#TextBild)
* [Headline](#Headline)
* [ParallaxSection](#ParallaxSection)
* [Institutionen](#Institutionen)
* [InstitutionenWhite](#InstitutionenWhite)
* [KontaktTeaser](#KontaktTeaser)
* [MarqueePraxispartner](#MarqueePraxispartner)

### HeroStartseite

**Pfad:**  
/Components/Hero/HeroStartseite.tsx

**Styled-Component Pfad:**

/Components/UI/Hero/HeroStartseite.styled.tsx

**Beschreibung:**  
Die Komponente **HeroStartseite** wird verwendet, um einen großen Hero-Bereich auf der Startseite anzuzeigen, der als visuelles Highlight dient. Hier können wichtige Informationen, ein Bild sowie ein Call-to-Action-Button angezeigt werden. Diese Komponente ist besonders flexibel, da sie über verschiedene Props gesteuert wird, die Inhalte und Ausrichtung dynamisch anpassen.

Anpassbare Props:

* **imgSrc**: Der Pfad zum Bild, das im Hero-Bereich angezeigt wird. In diesem Fall verweist es auf ImageMaestro.
* **imgAlt**: Der Alternativtext für das Bild, der für Barrierefreiheit und SEO wichtig ist. Dieser Text beschreibt das Bild.
* **headline**: Die Hauptüberschrift, die in prominenter Schrift im Hero-Bereich angezeigt wird. Sie vermittelt die zentrale Botschaft.
* **description**: Eine kurze Beschreibung oder Unterüberschrift, die mehr Details oder Kontext zur Hauptüberschrift bietet.
* **linkTo**: Ein Link, der den Nutzer zu einer anderen Seite weiterleitet, wenn auf den Button geklickt wird.
* **button**: Der Text des Call-to-Action-Buttons, der Nutzer zum Klicken auffordert.
* **$objectPosition**: Eine CSS-Eigenschaft, die bestimmt, wie das Bild innerhalb seines Containers positioniert wird. Dies beeinflusst vor allem, welcher Teil des Bildes sichtbar ist.

**Inhaltliche Anpassung:**  
Um die Inhalte der **HeroStartseite**-Komponente anzupassen, kannst du einfach die entsprechenden Props ändern. Diese befinden sich in der Datei, in der die Komponente eingebunden ist (der Startseite). Änderungen an den Werten der Props (wie Texten, Bildern und Links) spiegeln sich direkt in der Darstellung auf der Webseite wider.

### Marquee Banner

**Pfad:**  
/Components/Hero/MarqueeBanner.tsx

**Styled-Component Pfad:**  
/Components/UI/Banner/MarqueeBanner.styled.tsx

**Beschreibung:**  
Die Komponente **MarqueeBanner** erzeugt einen horizontal scrollenden Bereich mit Text und Logos, der kontinuierlich von rechts nach links läuft. Diese Komponente nutzt die Bibliothek **react-fast-marquee**, die ein einfaches und performantes Marquee-Scrolling in React-Anwendungen ermöglicht. Sie kann flexibel angepasst werden, indem verschiedene Optionen über die Props von **react-fast-marquee** gesteuert werden, wie Geschwindigkeit, Hover-Pause und der Einsatz von Gradienten.

Verwendete Bibliothek:

* **react-fast-marquee**: Diese Bibliothek wird verwendet, um den Marquee-Effekt zu erzeugen. Im Gegensatz zu traditionellen Marquee-Implementierungen bietet **react-fast-marquee** eine flüssige Animation und eine reibungslose Performance auch bei hoher Geschwindigkeit oder vielen Inhalten. Sie ermöglicht zudem erweiterte Funktionen wie das Pausieren des Scrollens beim Hover oder das Ausschalten von Gradienten.

Anpassbare Props von react-fast-marquee:

* **gradient** (boolean): Steuert, ob ein Gradient-Effekt am Anfang und Ende des Marquees angezeigt wird. Standardmäßig ist dies deaktiviert, um den Effekt auszuschalten.
  + *Beispiel:* gradient={false}
* **speed** (number): Gibt die Geschwindigkeit des Marquees an. Der Wert in dieser Komponente ist auf 50 gesetzt, was ein schnelles Scrollen bewirkt.
  + Beispiel: speed={50}
* **pauseOnHover** (boolean): Bestimmt, ob das Scrollen pausiert, wenn der Benutzer mit der Maus über den Marquee fährt. In dieser Komponente ist diese Option aktiviert.
  + *Beispiel:* pauseOnHover={true}

**Inhaltliche Anpassung:**  
Die Texte und Logos, die innerhalb des Marquees angezeigt werden, können in der **MarqueeWrapper**-Struktur angepasst werden. Hier kannst du Texte und Bilder einfügen oder ersetzen, um den angezeigten Inhalt zu ändern. Die Flexibilität von **react-fast-marquee** ermöglicht es, weitere Anpassungen vorzunehmen, wie das Ändern der Geschwindigkeit oder das Hinzufügen eines Gradient-Effekts.

* **Textanpassung:**  
  Der Text innerhalb des Marquees kann in den <h3>-Tags angepasst werden. Hier kannst du beliebige Texte hinzufügen, die innerhalb des Marquees wiederholt werden sollen.
  + Beispiel:jsx  
    <h3>Extended Reality – Lehrerbildung in virtuellen Kontexten</h3>
* **Logoanpassung:**  
  Die Logos, die im Marquee angezeigt werden, können im **LogoContainer** und **StyledLogoSVG** angepasst oder durch andere SVG-Elemente oder Bilder ersetzt werden.

### BildText

**Pfad:**  
/Components/InfoText/BildText.tsx

**Styled-Component Pfad:**  
/Components/UI/InfoText/BildText.styled.tsx

**Beschreibung:**  
Die **BildText**-Komponente zeigt einen Abschnitt an, der aus einem Bild und einem dazugehörigen Textblock mit einer Überschrift, einer Beschreibung und einem Button besteht. Diese Komponente eignet sich hervorragend, um Inhalte auf der Webseite in einem klar strukturierten Layout zu präsentieren. Die Position des Bildes sowie der Text und der Button können dynamisch durch Props gesteuert werden.

**Anpassbare Props:**

* **imgSrc** (string): Der Pfad zur Bilddatei, die im Bildbereich der Komponente angezeigt wird.
* **imgAlt** (string): Der Alternativtext für das Bild, der für Barrierefreiheit und SEO wichtig ist. Dieser Text beschreibt das Bild.
* **headline** (string): Die Hauptüberschrift, die im Textbereich prominent dargestellt wird.
* **description** (string): Ein Beschreibungstext, der unter der Überschrift angezeigt wird, um mehr Details oder Kontext zu liefern.
* **linkTo** (string): Der Link, zu dem der Button den Benutzer führt.
* **button** (string): Der Text des Call-to-Action-Buttons.
* **$objectPosition** (optional, string): Definiert, wie das Bild innerhalb seines Containers positioniert wird (z.B. center, top, bottom).

**Inhaltliche Anpassung:** Um die Inhalte der **BildText**-Komponente zu ändern, können die entsprechenden Props angepasst werden. Diese befinden sich in der Datei, in der die Komponente verwendet wird, auf der Startseite. Änderungen an den Werten der Props (wie Bildquelle, Texte und Links) beeinflussen die Darstellung dynamisch.

### TextBild

**Pfad:**  
/Components/InfoText/TextBild.tsx

**Styled-Component Pfad:**  
/Components/UI/InfoText/TextBild.styled.tsx

**Beschreibung:**  
Die **TextBild**-Komponente ist wie die **BildText**-Komponente aufgebaut nur Seitenverkehrt.

### Headline

**Pfad:**  
/Components/UI/Headline.styled.tsx

**Beschreibung:**  
Die **Headline**-Komponente ist eine einfache Styled Component, die eine <h2>-Überschrift darstellt. Sie wird verwendet, um zentralisierte, gut formatierte Überschriften an verschiedenen Stellen der Webseite anzuzeigen. Die Komponente ist so gestaltet, dass sie sich an unterschiedliche Bildschirmgrößen anpasst und responsiv ist.

### ParallaxSection

**Styled-Component Pfad:**  
/Components/UI/Parallax/ParallaxSection.styled.tsx

**Custom-Hook Pfad:**

/Components/customHooks/useStickyScroll.tsx

**Beschreibung:**  
Die **ParallaxSection**-Komponente ist eine Styled Component, die eine Parallax-Sektion auf der Webseite darstellt. Sie ermöglicht es, einen Sticky-Scrolling-Effekt zu implementieren, sodass bestimmte Inhalte beim Scrollen an einer festen Position bleiben. Diese Komponente wird verwendet, um mehrere andere Komponenten, wie z.B. **Institutionen** oder **InstitutionenWhite**, innerhalb eines flexiblen Layouts darzustellen.

Die Komponente verwendet **forwardRef**, um den Zugriff auf den DOM-Referenzknoten zu ermöglichen. Dies ist besonders nützlich für den Sticky-Effekt, der mit JavaScript gesteuert werden kann.

**Integration von anderen Komponenten:**

Die **ParallaxSection**-Komponente dient als Container für andere Komponenten, wie z.B. **Institutionen** oder **InstitutionenWhite**. Diese Komponenten werden als **children** in der ParallaxSection angezeigt. Die Sticky-Eigenschaft der ParallaxSection sorgt dafür, dass der Inhalt beim Scrollen der Seite an einer festen Position bleibt, bis der Benutzer weit genug gescrollt hat.

**forwardRef:**

Durch die Verwendung von **forwardRef** wird der Referenzknoten an die Komponente weitergeleitet. Dies ist hilfreich, um die Komponente in Verbindung mit anderen Funktionen wie dem **useStickyScroll**-Hook zu verwenden, der den Sticky-Status überwacht.

useStickyScroll Hook

**Beschreibung:**  
Der Custom Hook **useStickyScroll** ist dafür verantwortlich, den "sticky"-Effekt für mehrere Referenzen (Refs) zu steuern. Er verwendet den **IntersectionObserver**, um zu erkennen, wann ein Element den Viewport betritt oder verlässt, und fügt in diesen Fällen dynamisch eine "sticky"-Klasse hinzu oder entfernt diese.

Dieser Hook wird verwendet, um mehrere **ParallaxSection**-Komponenten zu überwachen, sodass sie beim Scrollen am oberen Rand des Bildschirms fixiert bleiben.

### Institutionen

**Pfad:**  
/Components/Parallax/Institutionen.tsx

**Styled-Component Pfad:**  
/Components/UI/Parallax/Institutionen.styled.tsx

**Beschreibung:**  
Die **Institutionen**-Komponente wird verwendet, um Informationen über verschiedene Institutionen darzustellen. Sie zeigt ein Bild, eine Überschrift, eine Beschreibung, Logos von Institutionen und einen Call-to-Action-Button an. Diese Komponente ist flexibel und wird hauptsächlich in der **ParallaxSection** verwendet, um Informationen dynamisch und visuell ansprechend darzustellen.

**Anpassbare Props:**

* **imgSrc** (string): Der Pfad zur Bilddatei, die in der Komponente angezeigt wird.
* **imgAlt** (string): Alternativtext für das Bild, das aus Barrierefreiheits- und SEO-Gründen wichtig ist.
* **logoOneSrc** (string): Der Pfad zum ersten Logo der Institution.
* **logoOneAlt** (string): Alternativtext für das erste Logo.
* **logoTwoSrc** (string): Der Pfad zum zweiten Logo der Institution (optional, falls vorhanden).
* **logoTwoAlt** (string): Alternativtext für das zweite Logo (falls vorhanden).
* **headline** (JSX.Element | string): Die Hauptüberschrift der Institution. Kann entweder ein JSX-Element oder ein einfacher Text sein.
* **description** (string): Der Beschreibungstext für die Institution.
* **linkTo** (string): Der Link, der den Benutzer zu einer weiteren Informationsseite führt.
* **button** (string): Der Text des Call-to-Action-Buttons.
* **$objectPosition** (optional, string): Definiert, wie das Bild innerhalb seines Containers positioniert wird (z.B. center, top, bottom).
* **$logoHeight** (optional, string): Legt die Höhe der Logos fest. Der Standardwert ist "40px".
* **$buttonVariant** (optional, "secondary"): Bestimmt das Aussehen des Buttons (z.B. sekundärer Stil).

**Verwendung der Institutionen-Komponente in der ParallaxSection:**

Die **Institutionen**-Komponente wird innerhalb der **ParallaxSection** verwendet, um visuell ansprechende Abschnitte darzustellen. Die Sticky-Scroll-Funktion der **ParallaxSection** lässt den Inhalt der **Institutionen**-Komponente dynamisch beim Scrollen hervortreten.

**Inhaltliche Anpassung:**  
Um die Inhalte der **Institutionen**-Komponente anzupassen, kannst du einfach die entsprechenden Props ändern. Diese befinden sich in der Datei, in der die Komponente eingebunden ist (der Startseite). Änderungen an den Werten der Props (wie Texten, Bildern und Links) spiegeln sich direkt in der Darstellung auf der Webseite wider.

### InstitutionenWhite

**Pfad:**  
/Components/Parallax/InstitutionenWhite.tsx

**Styled-Component Pfad:**  
/Components/UI/Parallax/InstitutionenWhite.styled.tsx

**Beschreibung:**  
Die **InstitutionenWhite**-Komponente ist wie die **Institutionen**-Komponente aufgebaut nur mit weißer Hintergrundfarbe.

### KontaktTeaser

**Pfad:**  
/Components/InfoText/KontaktTeaser.tsx

**Styled-Component Pfad:**  
/Components/UI/InfoText/KontaktTeaser.styled.tsx

**Beschreibung:**  
Die **KontaktTeaser**-Komponente wird verwendet, um eine Überschrift, zwei Beschreibungstexte und einen Call-to-Action-Button anzuzeigen. Diese Komponente ist besonders nützlich, um den Benutzer zu einem Kontaktformular oder einer anderen Seite mit Kontaktinformationen zu führen. Sie unterstützt optional auch den Dark Mode, der über den Zustand der Anwendung gesteuert wird.

Props:

* **headline** (string): Die Überschrift, die in der Komponente angezeigt wird.
* **descriptionOne** (string): Der erste Beschreibungstext, der den Benutzern zusätzliche Informationen liefert.
* **descriptionTwo** (string): Der zweite Beschreibungstext, der weitere Details vermittelt.
* **linkTo** (string): Der Link, zu dem der Button führen soll.
* **button** (string): Der Text, der auf dem Button angezeigt wird.
* **$buttonVariant** (optional, "tertiary"): Eine optionale Prop, um den Stil des Buttons zu ändern. Der Standardwert ist "tertiary", was dem Button ein spezifisches Design gibt.

**Inhaltliche Anpassung:**  
Um die Inhalte der **KontaktTeaser**-Komponente anzupassen, kannst du einfach die entsprechenden Props ändern. Diese befinden sich in der Datei, in der die Komponente eingebunden ist (zb. der Startseite). Änderungen an den Werten der Props (wie Texten und Links) spiegeln sich direkt in der Darstellung auf der Webseite wider.

### MarqueePraxispartner

**Pfad:**  
/Components/Banner/MarqueePraxispartner.tsx

**Styled-Component Pfad:**  
/Components/UI/Banner/MarqueePraxispartner.styled.tsx

**Beschreibung:**  
Die **MarqueePraxispartner**-Komponente erzeugt einen horizontal scrollenden Bereich mit Logos der Praxispartner, der kontinuierlich von rechts nach links läuft. Diese Komponente nutzt die Bibliothek **react-fast-marquee**, die ein einfaches und performantes Marquee-Scrolling in React-Anwendungen ermöglicht. Sie zeigt Logos von verschiedenen Institutionen, die an dem Projekt beteiligt sind, und bietet dabei eine reibungslose Animation. Über verschiedene Props der Bibliothek können Optionen wie Geschwindigkeit, Hover-Pause und der Gradient-Effekt angepasst werden.

**Verwendete Bibliothek:**

* **react-fast-marquee**: Diese Bibliothek wird verwendet, um den Marquee-Effekt zu erzeugen. Sie bietet eine flüssige Animation und gute Performance auch bei hoher Geschwindigkeit oder vielen Logos. Zudem ermöglicht sie erweiterte Funktionen wie das Pausieren des Scrollens beim Hover oder das Ausschalten von Gradienten.

**Anpassbare Props von react-fast-marquee:**

* **gradient** (boolean): Steuert, ob ein Gradient-Effekt am Anfang und Ende des Marquees angezeigt wird. In dieser Komponente ist der Effekt deaktiviert.
  + *Beispiel:* gradient={false}
* **speed** (number): Gibt die Geschwindigkeit des Marquees an. In dieser Komponente ist der Wert auf 50 gesetzt, was eine schnelle Bewegung bewirkt.
  + Beispiel: speed={50}
* **pauseOnHover** (boolean): Bestimmt, ob das Scrollen pausiert, wenn der Benutzer mit der Maus über den Marquee fährt. Diese Option ist in der Komponente aktiviert.
  + *Beispiel:* pauseOnHover={true}

**Inhaltliche Anpassung:**

Die Logos, die innerhalb des Marquees angezeigt werden, können im **ImageContainer** strukturiert werden. Hier können die Bilder (Logos) und Links hinzugefügt oder ausgetauscht werden, um den angezeigten Inhalt zu ändern. Die Logos werden in einem fortlaufenden Marquee animiert und wiederholen sich, um einen nahtlosen Scroll-Effekt zu gewährleisten.

## Termine Seite

Die **Termineseite** dient dazu, alle zukünftigen Fortbildungstermine anzukündigen und auf weitere Informationen zu verweisen.

Verwendete Hauptkomponenten:

* [HeroProjekt](#HeroProjekt)
* TerminBildText
* TerminTextBild

### HeroProjekt

**Pfad:**  
/Components/Hero/HeroProjekt.tsx

**Styled-Component Pfad:**  
/Components/UI/Hero/HeroProjekt.styled.tsx

**Beschreibung:**  
Die **HeroProjekt**-Komponente bildet den Einstieg in die Terminseite, ähnlich wie auf der Projektseite. Sie zeigt ein großes Bild mit einer zentralen Überschrift und einer kurzen Beschreibung, um den Besucher in das Thema einzuführen. Das Bild und die Texte können dynamisch über die bereitgestellten Props angepasst werden.

**Anpassbare Props:**

* **$imgSrc** (string): Der Pfad zum Hintergrundbild, das im Hero-Bereich angezeigt wird.
* **imgAlt** (string): Alternativtext für das Bild, der für Barrierefreiheit und SEO wichtig ist.
* **headline** (string): Die Hauptüberschrift, die in prominenter Schrift im Hero-Bereich angezeigt wird.
* **description** (string): Eine kurze Beschreibung oder Unterüberschrift, die mehr Details oder Kontext zur Hauptüberschrift bietet.
* **linkTo** (string): Ein Link, der den Nutzer zu einer weiteren Seite weiterleitet.
* **button** (string): Der Text des Call-to-Action-Buttons, der Nutzer zum Klicken auffordert.
* **$objectPosition** (string): Eine CSS-Eigenschaft, die bestimmt, wie das Bild innerhalb seines Containers positioniert wird.

**Inhaltliche Anpassung:**  
Die Inhalte der **HeroProjekt**-Komponente können durch die Anpassung der Props verändert werden. Dies betrifft das Bild, die Überschrift, die Beschreibung und den Button. Diese Anpassungen werden in der Datei vorgenommen, in der die Komponente aufgerufen wird -> zb. ProjektPage.tsx.

### TerminTextBild / TerminBildText

**Pfad:**  
/Components/InfoText/TerminTextBild.tsx bzw. TerminBildText.tsx

**Styled-Component Pfad:**  
/Components/UI/InfoText/TextBild.styled.tsx

**Beschreibung:**  
Die **TerminTextBild**-/**TerminBildText-**Komponenten sind abgeleitet von der **TextBild- / BildText-**Komponente. Sie zeigen einen Abschnitt an, der aus einem Bild und einem dazugehörigen Textblock mit einer Überschrift, einer Beschreibung und einem Button besteht. Diese Komponente eignet sich hervorragend, um Inhalte auf der Webseite in einem klar strukturierten Layout zu präsentieren. Die Position des Bildes sowie der Text und der Button können dynamisch durch Props gesteuert werden.

Die **TerminTextBild**-Komponente ist wie die **TerminBildText**-Komponente aufgebaut nur Seitenverkehrt.

**Anpassbare Props:**

* **imgSrc** (string): Der Pfad zur Bilddatei, die im Bildbereich der Komponente angezeigt wird.
* **imgAlt** (string): Der Alternativtext für das Bild, der für Barrierefreiheit und SEO wichtig ist. Dieser Text beschreibt das Bild.
* **headline** (string): Die Hauptüberschrift, die im Textbereich prominent dargestellt wird.
* **date** (string): Das Datum der Veranstaltung
* **description** (string): Ein Beschreibungstext, der unter der Überschrift angezeigt wird, um mehr Details oder Kontext zu liefern.
* **linkTo** (string): Der Link, zu dem der Button den Benutzer führt.
* **button** (string): Der Text des Call-to-Action-Buttons.
* **$objectPosition** (optional, string): Definiert, wie das Bild innerhalb seines Containers positioniert wird (z.B. center, top, bottom).

**Besonderheiten:**

Der Button wird nur dargestellt, wenn der linkTo-String nichtleer ist.

**Inhaltliche Anpassung:** Um die Inhalte der **TerminTextBild**-/**TerminBildText-**Komponenten zu ändern, können die entsprechenden Props angepasst werden. Diese befinden sich in der Datei, in der die Komponente verwendet wird, auf der Startseite. Änderungen an den Werten der Props (wie Bildquelle, Texte und Links) beeinflussen die Darstellung dynamisch.

## Projekt Seite

Die **Projektseite** besteht aus verschiedenen Hauptkomponenten, die zur Struktur und Gestaltung der Seite beitragen. Jede Komponente hat eine spezifische Funktion und trägt dazu bei, die Ziele und Inhalte des Projekts „LEVIKO-XR" zu kommunizieren.

Verwendete Hauptkomponenten:

* [HeroProjekt](#HeroProjekt)
* [ProjektAccordion](#ProjektAccordion)
* [KontaktTeaser](#KontaktTeaser)
* [Headline](#Headline)
* [ParallaxSection](#ParallaxSection)
* [ProjektInfoParallax](#ProjektInfoParallax)
* [ProjektInfoParallaxWhite](#ProjektInfoParallaxWhite)

### HeroProjekt

**Pfad:**  
/Components/Hero/HeroProjekt.tsx

**Styled-Component Pfad:**  
/Components/UI/Hero/HeroProjekt.styled.tsx

**Beschreibung:**  
Die **HeroProjekt**-Komponente bildet den Einstieg in die Projektseite. Sie zeigt ein großes Bild mit einer zentralen Überschrift und einer kurzen Beschreibung, um den Besucher in das Thema einzuführen. Das Bild und die Texte können dynamisch über die bereitgestellten Props angepasst werden.

**Anpassbare Props:**

* **$imgSrc** (string): Der Pfad zum Hintergrundbild, das im Hero-Bereich angezeigt wird.
* **imgAlt** (string): Alternativtext für das Bild, der für Barrierefreiheit und SEO wichtig ist.
* **headline** (string): Die Hauptüberschrift, die in prominenter Schrift im Hero-Bereich angezeigt wird.
* **description** (string): Eine kurze Beschreibung oder Unterüberschrift, die mehr Details oder Kontext zur Hauptüberschrift bietet.
* **linkTo** (string): Ein Link, der den Nutzer zu einer weiteren Seite weiterleitet.
* **button** (string): Der Text des Call-to-Action-Buttons, der Nutzer zum Klicken auffordert.
* **$objectPosition** (string): Eine CSS-Eigenschaft, die bestimmt, wie das Bild innerhalb seines Containers positioniert wird.

**Inhaltliche Anpassung:**  
Die Inhalte der **HeroProjekt**-Komponente können durch die Anpassung der Props verändert werden. Dies betrifft das Bild, die Überschrift, die Beschreibung und den Button. Diese Anpassungen werden in der Datei vorgenommen, in der die Komponente aufgerufen wird -> zb. ProjektPage.tsx.

### ProjektAccordion

**Pfad:**  
/Components/Accordion/ProjektAccordion.tsx

**Data Pfad:**  
/Components/Accordion/ProjektAccordionData.tsx

**Styled-Component Pfad:**  
/Components/UI/Accordion/ProjektAccordion.styled.tsx

**Store Pfad:**  
/Components/stores/useAccordionStore.tsx

**Beschreibung:**  
Die **ProjektAccordion**-Komponente bietet eine strukturierte Möglichkeit, Informationen über das Projekt in einem Akkordeon-Stil anzuzeigen. Jede Sektion enthält eine Überschrift und einen ausklappbaren Inhalt. Diese Komponente verwendet die Bibliothek **react-accessible-accordion** für die Akkordeon-Funktionalität und ist stark anpassbar. Das Styling wird über **Styled Components** implementiert. Zusätzlich wird der Zustand des ausgeklappten Akkordeons über einen **Zustand Store** mit **Zustand** verwaltet, der speichert, welche Sektion gerade geöffnet ist.

**Verwendete Bibliotheken:**

* **react-accessible-accordion**: Diese Bibliothek stellt die Akkordeon-Funktionalität zur Verfügung. Sie sorgt dafür, dass die Akkordeon-Elemente zugänglich und kompatibel mit Bildschirmlesegeräten sind.
* **Zustand**: Ein einfaches Zustand-Management-Tool, das hier verwendet wird, um den aktuellen Zustand des Akkordeons zu speichern und zu verwalten.

**Anpassbare Props:**

* **items** (Array): Ein Array, das die Akkordeon-Elemente definiert. Jedes Element enthält:
  + **id**: Eindeutige ID des Akkordeon-Elements, um den Zustand korrekt zu verwalten.
  + **title**: Die Überschrift des Akkordeon-Elements.
  + **content**: Der Inhalt des Akkordeon-Elements, der ein JSX-Element oder einfacher Text sein kann.

**Inhaltliche Anpassung:**

Um die Inhalte des Akkordeons anzupassen, kann das **items**-Array -> ProjektAccordionData.tsx <- modifiziert werden. Jede Akkordeon-Sektion wird über dieses Array definiert und kann einfach um neue Einträge erweitert oder bestehende Einträge können geändert werden.

### Verwendete Components

**Beschreibung:**

Die folgenden 3 Components wurden bereits auf der Startseite dokumentiert.

* [KontaktTeaser](#KontaktTeaser)
* [Headline](#Headline)
* [ParallaxSection](#ParallaxSection)

### ProjektInfoParallax

**Pfad:**  
/Components/Parallax/ProjektInfoParallax.tsx

**Styled-Component Pfad:**  
/Components/UI/Parallax/ProjektInfoParallax.styled.tsx

**Beschreibung:**  
Die **ProjektInfoParallax**-Komponente wird verwendet, um Informationen zu einem bestimmten Thema im Parallax-Stil darzustellen. Sie kombiniert eine Überschrift, ein Bild, eine Beschreibung sowie einen Call-to-Action-Button. Diese Komponente ist flexibel und wird hauptsächlich in der **ParallaxSection** verwendet, um den Inhalt dynamisch und visuell ansprechend darzustellen. Sie eignet sich besonders gut für das Hervorheben von Projektinformationen oder wichtigen Themen.

**Anpassbare Props:**

* **headline** (string): Die Überschrift des Parallax-Bereichs, die den Hauptinhalt des Abschnitts darstellt.
* **imgSrc** (string): Der Pfad zur Bilddatei, die in der Komponente angezeigt wird.
* **imgAlt** (string): Alternativtext für das Bild, der für Barrierefreiheit und SEO wichtig ist.
* **$objectPosition** (optional, string): Definiert die Positionierung des Bildes innerhalb seines Containers. Der Standardwert ist "center".
* **linkTo** (string): Der Link, der den Benutzer zu einer weiteren Seite oder zu mehr Informationen führt.
* **button** (string): Der Text des Call-to-Action-Buttons, der den Benutzer zu weiteren Inhalten führt.
* **$buttonVariant** (optional, "secondary"): Optionale Prop, um den Stil des Buttons anzupassen. Der Wert "secondary" gibt dem Button einen alternativen Stil.
* **description** (JSX.Element | string): Der Beschreibungstext, der als JSX-Element oder einfacher Text übergeben werden kann. Dieser Text bietet zusätzliche Informationen zum Thema.

**Verwendung der ProjektInfoParallax-Komponente in der ParallaxSection:**

Die **ProjektInfoParallax**-Komponente wird innerhalb der **ParallaxSection** verwendet, um ansprechende visuelle Inhalte mit einem fließenden Scroll-Effekt darzustellen. Dieser Parallax-Abschnitt lässt den Inhalt dynamisch hervortreten, während der Benutzer durch die Seite scrollt. Das Bild und der Text wechseln sich ab, um den Lesefluss und die visuelle Anziehungskraft zu maximieren.

**Inhaltliche Anpassung:**

Um die Inhalte der **ProjektInfoParallax**-Komponente anzupassen, können die entsprechenden Props geändert werden. Die Komponente ist so gestaltet, dass Änderungen an den Props (wie Texten, Bildern und Links) direkt in der Darstellung auf der Webseite reflektiert werden. Diese Anpassungen werden in der Datei vorgenommen, in der die Komponente aufgerufen wird -> Projekt.tsx.

### ProjektInfoParallaxWhite

**Pfad:**  
/Components/Parallax/ProjektInfoParallaxWhite.tsx

**Styled-Component Pfad:**  
/Components/UI/Parallax/ProjektInfoParallaxWhite.styled.tsx

**Beschreibung:**  
Die **ProjektInfoParallaxWhite**-Komponente ist wie die **ProjektInfoParallax**-Komponente aufgebaut nur mit weißer Hintergrundfarbe.

## Forschungsseite

Die Forschungsseite besteht aus verschiedenen Hauptkomponenten, die zur Struktur und Gestaltung der Seite beitragen. Jede Komponente hat eine spezifische Funktion und unterstützt die Darstellung der Forschungserkenntnisse und -methoden des Projekts „LEVIKO-XR“. Die Seite bietet eine detaillierte Übersicht der Forschungsfragen, Methoden und Modelle, die im Rahmen des Projekts verwendet werden. Jedes Diagramm fängt mit DiagrammIntroText oder DiagrammIntro an und endet mit DiagrammOutro.

Verwendete Hauptkomponenten:

* [HeroProjekt](#HeroProjekt)
* [ForschungNav](#ForschungNav)
* [Headline](#Headline)
* [ParallaxSection](#ParallaxSection)
* [Forschungsfragen](#Forschungsfragen)
* [ForschungsfragenWhite](#ForschungsfragenWhite)
* [KontaktTeaser](#KontaktTeaser)
* [MarqueeNews](#MarqueeNews)
* [DiagrammIntroText](#DiagrammIntroText)
* [DiagrammIntro](#DiagrammIntro)
* [DiagrammOutro](#DiagrammOutro)
* [DBRDiagramm](#DBRDiagramm)
* [TPACKDiagramm](#TPACKDiagramm)
* [SAMRDiagramm](#SAMRDiagramm)
* [CAMILDiagramm](#CAMILDiagramm)
* [KerresDiagramm](#KerresDiagramm)

### ForschungNav

**Pfad:**  
/Components/InfoText/ForschungNav.tsx

**Styled-Component Pfad:**  
/Components/UI/InfoText/ForschungNav.styled.tsx

**Beschreibung:**  
Die **ForschungNav**-Komponente dient als Navigationselement auf der Forschungsseite. Sie ermöglicht es dem Benutzer, schnell zu den wichtigsten Abschnitten der Seite zu scrollen, wie **Forschungsfragen**, **Methoden** und **Modelle**. Darüber hinaus bietet sie einen Teaser für die Publikationsseite, auf die der Benutzer über einen Call-to-Action-Link weitergeleitet werden kann. Die Komponente ist responsiv und unterstützt den **Dark Mode**, der den Stil dynamisch anpasst.

**Anpassbare Props:**

* **scrollToSection(id: string)**: Eine Funktion, die es dem Benutzer ermöglicht, durch Anklicken der Navigationspunkte zu einem bestimmten Abschnitt auf der Seite zu springen. Die ID des Zielabschnitts wird als Parameter übergeben.
* **isDarkModeOn (boolean)**: Der Zustand, der bestimmt, ob der Dark Mode aktiviert ist. Die Komponente passt basierend auf diesem Zustand Farben und Icons an.

**Inhaltliche Elemente:**

* **NavHeadline (string)**: Die Überschrift der Navigation.
* **NavP**: Der Text für die verschiedenen Navigationspunkte. Diese Texte sind anklickbar und rufen die **scrollToSection**-Funktion auf, um den Benutzer zu den jeweiligen Abschnitten der Seite zu navigieren. Je nach Dark Mode wird die Farbe der Texte angepasst.
* **PublikationsTeaser (string)**: Ein kurzer Teaser, der den Benutzer dazu ermutigt, mehr über die Forschungsarbeit zu erfahren. Dieser Bereich enthält einen Call-to-Action-Link, der zur Publikationsseite führt.
* **Link (React Router)**: Der Call-to-Action-Link, der den Benutzer zur **Publikationen**-Seite weiterleitet.

**Inhaltliche Anpassung:**  
Die Inhalte der **ForschungNav**-Komponente können leicht durch die Änderung der Texte, der Links und der IDs der Zielabschnitte angepasst werden. Die Navigationspunkte (z.B. **Forschungsfragen**, **Methode**, **Modelle**) sind anklickbar und führen zu den entsprechenden Bereichen auf der Seite. Zusätzlich kann der Publikationsteaser und der Call-to-Action-Link an die Bedürfnisse des Projekts angepasst werden.

### Forschungsfragen

**Pfad:**  
/Components/Parallax/Forschungsfragen.tsx

**Styled-Component Pfad:**  
/Components/UI/Parallax/Forschungsfragen.styled.tsx

**Beschreibung:**  
Die **Forschungsfragen**-Komponente wird verwendet, um zentrale Forschungsfragen im Parallax-Stil darzustellen. Sie zeigt eine Überschrift, die entweder als einfacher Text oder als JSX-Element übergeben werden kann, und wird hauptsächlich in der **ParallaxSection** genutzt, um dynamische und visuell ansprechende Inhalte zu präsentieren. Diese Komponente eignet sich besonders gut, um wichtige Fragen und Schwerpunkte der Forschungsarbeit hervorzuheben.

Anpassbare Props:

* **headline (JSX.Element | string)**: Die Hauptüberschrift des Parallax-Bereichs. Diese kann entweder als einfacher Text oder als JSX-Element übergeben werden, um größere Flexibilität bei der Formatierung zu ermöglichen.

**Inhaltliche Anpassung:**

Die Inhalte der **Forschungsfragen**-Komponente können durch die Änderung der **headline**-Prop angepasst werden. Diese Anpassungen werden in der Datei vorgenommen, in der die Komponente aufgerufen wird -> Forschung.tsx.

### ForschungsfragenWhite

**Pfad:**  
/Components/Parallax/ForschungsfragenWhite.tsx

**Styled-Component Pfad:**  
/Components/UI/Parallax/ForschungsfragenWhite.styled.tsx

**Beschreibung:**  
Die **ForschungsfragenWhite**-Komponente ist wie die **Forschungsfragen**-Komponente aufgebaut nur mit weißer Hintergrundfarbe.

### MarqueeNews

**Pfad:**

* /Components/Banner/MarqueeNews.tsx

**Styled-Component Pfad:**

* /Components/UI/Banner/MarqueeNews.styled.tsx

**Beschreibung:**

Die MarqueeNews-Komponente zeigt eine fortlaufend scrollende Liste von bevorstehenden Veranstaltungen an, bei denen LEVIKO-XR teilnimmt. Sie verwendet das react-fast-marquee-Paket, um ein horizontales Laufband zu erzeugen, das automatisch scrollt. Zusätzlich zeigt sie am unteren Rand einen Text, der auf die Teilnahme von LEVIKO-XR an diesen Events hinweist.

**Anpassbare Features:**

* Marquee-Effekte:
  + **Scroll-Geschwindigkeit:** Die Scroll-Geschwindigkeit kann über die speed-Prop des react-fast-marquee-Pakets angepasst werden. In dieser Komponente ist die Geschwindigkeit auf 50 gesetzt.
  + **Pause-on-Hover:** Das Laufband pausiert, wenn der Benutzer mit der Maus darüber fährt (pauseOnHover={true}), was eine benutzerfreundlichere Interaktion ermöglicht.
  + **Gradient:** Es wurde auf das Standard-Gradient des Marquees verzichtet (gradient={false}), um ein sauberes und durchgehendes Design zu gewährleisten.
* Verlinkungen zu Veranstaltungen:
  + **LinkContainer:** Jeder Eventeintrag enthält einen klickbaren Link, der den Benutzer zu einer externen Seite mit mehr Informationen über die jeweilige Veranstaltung weiterleitet.
  + **Event-Texte:** Die Veranstaltungstexte sind dynamisch und können direkt in der Komponente geändert werden.

**Inhaltliche Anpassung:**

Die Inhalte der MarqueeNews-Komponente können durch die Änderung der Veranstaltungslinks und -texte im MarqueeWrapper angepasst werden. Die LinkContainer-Elemente ermöglichen es, Text für jede Veranstaltung und ihre jeweiligen URLs zu definieren. Das untere Textfeld im TextContainer ist ebenfalls frei anpassbar und bietet eine zusätzliche Möglichkeit, Informationen über die Teilnahme an den Events hinzuzufügen.

### DiagrammIntroText

**Pfad:**  
/Components/Diagramme/DiagrammIntroText.tsx

**Styled-Component Pfad:**  
/Components/UI/Diagramme/DiagrammeIntroText.styled.tsx

**Beschreibung:**  
Die **DiagrammIntroText**-Komponente dient dazu, eine kurze Einleitung für den nachfolgenden Abschnitt oder ein Diagramm bereitzustellen. Sie besteht aus einer Überschrift und einem Beschreibungstext, die beide flexibel als einfacher Text oder als JSX-Element übergeben werden können. Die Komponente wird oft verwendet, um methodische Ansätze oder Erklärungen vor Diagrammen einzuleiten.

**Anpassbare Props:**

* **headline (JSX.Element | string)**: Die Hauptüberschrift, die vor einem Diagramm oder einem methodischen Ansatz angezeigt wird. Sie kann als einfacher String oder als JSX-Element übergeben werden, um flexiblere Formatierungen zu ermöglichen.
* **description (JSX.Element | string)**: Der Beschreibungstext, der eine kurze Erklärung oder einen Kontext zur Überschrift bietet. Auch dieser kann als String oder JSX-Element übergeben werden, um komplexere Inhalte zu ermöglichen.

**Inhaltliche Anpassung:**

Die Inhalte der **DiagrammIntroText**-Komponente können durch die Änderung der **headline**- und **description**-Props angepasst werden. Diese ermöglichen es, sowohl einfache Texte als auch komplexere Inhalte mit HTML-Formatierungen darzustellen. Diese Anpassungen werden in der Datei vorgenommen, in der die Komponente aufgerufen wird -> Forschung.tsx.

### DiagrammIntro

**Pfad:**

* /Components/Diagramme/DiagrammIntro.tsx

**Styled-Component Pfad:**

* /Components/UI/Diagramme/DiagrammIntro.styled.tsx

**Beschreibung:**

Die DiagrammIntro-Komponente dient dazu, eine Einleitung zu einem Diagramm oder methodischen Abschnitt zu geben. Sie besteht aus einer Überschrift, einer Beschreibung und optional verschiedenen SVG-Diagrammen, die basierend auf dem displayedSVG-Prop dynamisch angezeigt werden können. Die Komponente wird häufig verwendet, um visuelle Erklärungen und Einleitungen für methodische Ansätze bereitzustellen.

**Anpassbare Props:**

* **headline (JSX.Element | string):** Die Hauptüberschrift der Komponente. Sie kann entweder als einfacher Text oder als JSX-Element übergeben werden, um eine flexible Formatierung zu ermöglichen.
* **description (JSX.Element | string):** Der Beschreibungstext, der den Kontext oder eine kurze Erklärung zur Überschrift bietet. Auch dieser kann als einfacher Text oder als JSX-Element übergeben werden.
* **displayedSVG (optional, string):** Ein optionales Prop, das das angezeigte SVG innerhalb der Komponente steuert. Es gibt drei mögliche Optionen:
  + "box": Zeigt ein rechteckiges Diagramm an.
  + "tpack": Zeigt das TPACK-Diagramm an, das spezifisch für den TPACK-Kontext verwendet wird.
  + "circle": Zeigt ein kreisförmiges Diagramm an.
* Der Standardwert ist "box", falls keine spezifische Auswahl getroffen wird.

**Inhaltliche Anpassung:**

Die Inhalte der DiagrammIntro-Komponente können durch die Anpassung der headline- und description-Props geändert werden. Der angezeigte Diagrammtyp kann über das displayedSVG-Prop gesteuert werden, wodurch verschiedene SVG-Elemente wie Boxen, Kreise oder das TPACK-Diagramm gerendert werden können.

### DiagrammOutro

**Pfad:**

* /Components/Diagramme/MethodenOutro.tsx

**Styled-Component Pfad:**

* /Components/UI/Diagramme/MethodenOutro.styled.tsx

**Beschreibung:**

Die MethodenOutro-Komponente wird verwendet, um eine kurze Beschreibung sowie einen Button anzuzeigen, der den Nutzer zu einer anderen Seite oder einem anderen Abschnitt weiterleitet. Sie eignet sich gut als abschließender Abschnitt nach methodischen Erklärungen oder Diagrammen, der den Nutzer zu weiteren Informationen oder Aktionen auffordert.

**Anpassbare Props:**

* description (JSX.Element | string):
  + Der Beschreibungstext oder die Zusammenfassung, die in der Komponente angezeigt wird. Der Inhalt kann entweder als einfacher String oder als JSX-Element übergeben werden, was eine flexible Gestaltung ermöglicht.
* linkTo (string):
  + Der Ziel-Link, zu dem der Button navigiert. Dies kann ein interner oder externer Link sein, abhängig von der App-Routing-Struktur.
* button (string):
  + Der Text, der auf dem Button angezeigt wird. Dieser sollte eine klare Handlungsaufforderung darstellen, wie z.B. "Mehr erfahren" oder "Zur Methode“.

**Inhaltliche Anpassung:**

Die Inhalte der MethodenOutro-Komponente können durch das Ändern der description- und button-Props angepasst werden. Die description kann einfachen Text oder auch komplexere Inhalte mit HTML-Elementen umfassen. Der Button kann durch das Setzen eines anderen linkTo-Werts an eine beliebige Seite weiterleiten.

### DBRDiagramm

**Pfad:**  
/Components/Diagramme/DBRDiagramm.tsx

**SVG Pfad:**  
/Components/UI/SVG/DBRsvg.tsx

**Styled-Component Pfad:**  
/Components/UI/Diagramme/DBRDiagramm.styled.tsx

**Store Pfad:**  
/Components/stores/useDBRStore.tsx

**Beschreibung:**  
Die **DBRDiagramm**-Komponente visualisiert den **Design-Based Research (DBR)**-Prozess. Dieser Ansatz besteht aus mehreren iterativen Phasen, die als interaktives Diagramm dargestellt werden. Durch Klicken auf die verschiedenen Elemente des Diagramms (z.B. Kreise oder Rechtecke) werden weitere Informationen zu den jeweiligen Phasen angezeigt. Die Komponente nutzt einen Zustand Store (Zustand-Management), um zu speichern, welche Phase des DBR-Prozesses ausgewählt wurde, und passt den angezeigten Inhalt dynamisch an.

**Verwendete Bibliotheken:**

* **Zustand:**  
  Ein einfaches und flexibles Zustand-Management-Tool, das verwendet wird, um den aktuell ausgewählten Abschnitt des DBR-Diagramms zu verwalten. Die ausgewählte Phase (z.B. „Requirements“, „Design“, „Analysis“) wird im Zustand Store gespeichert, um den angezeigten Inhalt basierend auf der Benutzerinteraktion zu ändern.

**Anpassbare Props:**

Es gibt keine direkten Props, die an die **DBRDiagramm**-Komponente übergeben werden. Stattdessen erfolgt die Anpassung über den Zustand Store und die **DBRSVG**-Komponente.

Funktionalitäten:

* **selectedBox (string)**: Diese Variable im Zustand speichert die aktuell ausgewählte Phase des DBR-Prozesses. Durch Klicken auf die jeweiligen Diagramm-Elemente wird die ausgewählte Phase geändert.
* **handleCircleClick(boxId: string)**: Diese Funktion wird aufgerufen, wenn ein Kreis oder ein Rechteck im Diagramm angeklickt wird. Sie ändert die aktuell ausgewählte Phase im Zustand.
* **isMobile (boolean)**: Der Zustand prüft, ob das Diagramm auf einem mobilen Gerät angezeigt wird (Fensterbreite ≤ 1024px). Je nach Gerätetyp wird das Layout des Diagramms entsprechend angepasst.

**Inhaltliche Anpassung:**

Die Inhalte der DBR-Phasen werden im **useDBRStore** definiert. Hier kann für jede Phase des Diagramms die Überschrift und die Beschreibung angepasst werden. Der Inhalt wird über die **contentMap** im Zustand Store verwaltet.

### TPACKDiagramm

**Pfad:**  
/Components/Diagramme/TPACKDiagramm.tsx

**SVG Pfad:**  
/Components/UI/SVG/TPACKsvg.tsx

**Styled-Component Pfad:**  
/Components/UI/Diagramme/TPACKDiagramm.styled.tsx

**Store Pfad:**  
/Components/stores/useTPACKStore.tsx

**Beschreibung:**

Die TPACKDiagramm-Komponente visualisiert das TPACK-Rahmenmodell (Technological Pedagogical Content Knowledge) in einer interaktiven Form. Nutzer können auf verschiedene Bereiche des Diagramms klicken, um zusätzliche Informationen zu den jeweiligen Komponenten zu erhalten. Diese Komponente verwendet Zustand-Management (Zustand) und ermöglicht eine dynamische Anpassung der Inhalte, abhängig von der Auswahl der Nutzer.

Das Diagramm verwendet das SVG-Element TPACKSVG, das die verschiedenen Bereiche des TPACK-Rahmenmodells darstellt. Die ausgewählten Bereiche ändern ihre Farbe, um die Interaktion visuell hervorzuheben, und die zugehörigen Informationen werden im Textbereich neben dem Diagramm angezeigt.

**Verwendete Bibliotheken:**

* **Zustand**: Wird verwendet, um den aktuell ausgewählten Pfad im TPACK-Diagramm und dessen zugehörige Informationen zu verwalten. Dies ermöglicht es, den Zustand des Diagramms und die angezeigten Inhalte dynamisch zu aktualisieren, wenn der Benutzer mit dem Diagramm interagiert.

**Anpassbare Inhalte:**

* **headline** (string): Überschrift des aktuell ausgewählten TPACK-Bereichs. Diese wird im Informationstext neben dem Diagramm angezeigt.
* **description** (string): Beschreibung des ausgewählten TPACK-Bereichs, die ebenfalls im Informationstext dargestellt wird.
* **selectedPath** (string | null): Der aktuell im Diagramm ausgewählte Pfad, der bestimmt, welche Informationen im Textbereich angezeigt werden.

**Interaktive Elemente:**

* Jeder Bereich des TPACK-Diagramms ist ein klickbares Element. Beim Anklicken eines Bereichs wie Technological Knowledge, Pedagogical Knowledge oder Content Knowledge ändern sich sowohl die Farbe des ausgewählten Bereichs als auch der Informationstext, um den Inhalt zu reflektieren.

**Inhaltliche Anpassung:**

Die Inhalte der TPACK-Diagramm-Komponente können über den useTPACKStore-Store angepasst werden. Der Store speichert die unterschiedlichen Pfad-IDs (path1, path2, etc.) und die dazugehörigen Inhalte wie headline und description.

Um die angezeigten Inhalte zu ändern, kann man im Zustand Store (in der Datei useTPACKStore.tsx) die entsprechenden headline- und description-Texte für jeden Pfad anpassen.

### SAMRDiagramm

**Pfad:**  
/Components/Diagramme/SAMRDiagramm.tsx

**Styled-Component Pfad:**  
/Components/UI/Diagramme/SAMRDiagramm.styled.tsx

**Store Pfad:**  
/Components/stores/useSAMRDStore.tsx

**Beschreibung:**

Die SAMRDiagramm-Komponente visualisiert das SAMR-Modell (Substitution, Augmentation, Modification, Redefinition) interaktiv. Es ermöglicht den Nutzern, auf verschiedene Abschnitte des Modells zu klicken und detailliertere Informationen über die jeweilige Stufe zu sehen. Die Inhalte der verschiedenen Stufen werden über einen Zustand-Store verwaltet, und die Darstellung passt sich dynamisch an, basierend auf der Benutzerinteraktion.

**Verwendete Bibliotheken:**

* **Zustand**: Ein Zustand-Management-Tool, das verwendet wird, um die ausgewählte Stufe im SAMR-Modell und die damit verbundenen Inhalte zu speichern und zu aktualisieren. Dies ermöglicht eine dynamische Anpassung der angezeigten Informationen, basierend auf der Auswahl der Nutzer.

**Anpassbare Inhalte:**

* **clickedId** (string | null): Speichert die ID des aktuell ausgewählten Modells (z.B. container1 für Substitution, container2 für Augmentation usw.).
* **content** (InfoBoxContent[]): Ein Array, das die verschiedenen SAMR-Stufen und ihre zugehörigen kurzen und ausführlichen Beschreibungen enthält. Jede Box besteht aus:
  + **id** (string): Die eindeutige ID des InfoContainers.
  + **title** (string): Der Titel der SAMR-Stufe (z.B. Substitution, Augmentation).
  + **shortText** (string): Kurze Beschreibung der Stufe.
  + **fullText** (string): Ausführliche Beschreibung der Stufe, die angezeigt wird, wenn die Stufe ausgewählt ist.

**Interaktive Elemente:**

* Die Nutzer können auf jeden Bereich des Modells klicken, um weitere Informationen über die jeweilige SAMR-Stufe zu erhalten. Wenn ein Bereich ausgewählt wird, ändert sich die Darstellung des Modells visuell, und der dazugehörige längere Text wird angezeigt.

**Inhaltliche Anpassung:**

Die Inhalte der SAMR-Diagramm-Komponente können über den useSAMRDStore angepasst werden. Das content-Array im Zustand-Store enthält die verschiedenen Informationen für jede SAMR-Stufe, die sowohl die kurze als auch die ausführliche Beschreibung jeder Stufe abdecken.

Um den Text oder die Titel der SAMR-Stufen zu ändern, kann der Entwickler die entsprechenden Einträge im content-Array im Zustand-Store modifizieren.

### CAMILDiagramm

**Pfad:**  
/Components/Diagramme/CAMILDiagramm.tsx

**Styled-Component Pfad:**  
/Components/UI/Diagramme/CAMILDiagramm.styled.tsx

**Store Pfad:**  
/Components/stores/useCAMILStore.tsx

**Beschreibung:**

Die CAMILDiagramm-Komponente stellt das CAMIL-Modell interaktiv dar, indem es fünf Schlüsselbereiche des Modells (Content, Audience, Media, Implementation, und Learning) visualisiert. Nutzer können durch das Klicken auf die verschiedenen Kreise detaillierte Informationen zu jedem Bereich des Modells anzeigen lassen.

Die Informationen und die Auswahl des aktuell aktiven Bereichs werden über den Zustand-Store (useCAMILStore) verwaltet, und die Darstellung wird dynamisch an die Nutzerinteraktion angepasst.

**Verwendete Bibliotheken:**

* **Zustand**: Wird verwendet, um den aktuell aktiven Bereich (activeCircle) und die zugehörigen Inhalte wie Titel und Beschreibung zu speichern. Durch diese Verwaltung kann die Auswahl und Anzeige von Inhalten effizient gesteuert werden.

**Anpassbare Inhalte:**

* **activeCircle** (number): Speichert den Index des aktuell ausgewählten Kreises. Dieser Index bestimmt, welche Informationen gerade angezeigt werden (z.B. Content, Audience, Media usw.).
* **title** (string): Der Titel des aktuell ausgewählten Bereichs des CAMIL-Modells.
* **content** (string): Der beschreibende Text, der zum ausgewählten Bereich angezeigt wird.

**Interaktive Elemente:**

* Die Nutzer können auf die verschiedenen Kreise des Diagramms klicken, um detaillierte Informationen zu den einzelnen Bereichen des CAMIL-Modells zu sehen. Jeder Kreis ist einem spezifischen Bereich zugeordnet:
  + **Content (Inhalt)**: Lehrplan oder Lernziele.
  + **Audience (Zielgruppe)**: Die Lernenden, für die der Unterricht gestaltet wird.
  + **Media (Medien)**: Die Technologien, die im Unterricht verwendet werden.
  + **Implementation (Implementierung)**: Wie die Technologien im Unterricht integriert werden.
  + **Learning (Lernen)**: Die Lernziele und wie sie durch Technologie verbessert werden.

Jeder Bereich wird durch ein Icon repräsentiert, und der Nutzer kann durch Klicks auf die Icons zwischen den Bereichen wechseln. Die Inhalte des aktiven Bereichs ändern sich dynamisch basierend auf der Auswahl.

**Inhaltliche Anpassung:**

Die Inhalte der CAMILDiagramm-Komponente können im useCAMILStore-Zustand-Store angepasst werden. Dieser Store enthält die Logik, um basierend auf dem ausgewählten Kreis (activeCircle) die richtigen Informationen anzuzeigen.

### KerresDiagramm

**Pfad:**  
/Components/Diagramme/KerresDiagramm.tsx

**Styled-Component Pfad:**  
/Components/UI/Diagramme/KerresDiagramm.styled.tsx

**Store Pfad:**  
/Components/stores/useKerresStore.tsx

**Beschreibung:**

Die **KerresDiagramm**-Komponente visualisiert das Rahmenmodell der Mediendidaktik von Kerres in Form einer interaktiven Boxen-Darstellung. Diese Darstellung fokussiert sich auf die technischen Merkmale, Erlebniselemente, Lernprozesse und Lernergebnisse, die in immersiven, virtuellen Umgebungen auftreten. Jede Box enthält spezifische Inhalte und durch Anklicken der Boxen können detaillierte Informationen zu den jeweiligen Aspekten des Modells angezeigt werden. Die Informationen zu den Boxen werden über einen Zustand-Store (useKerresStore) verwaltet und dynamisch angepasst.

**Verwendete Bibliotheken:**

* **Zustand**: Zustand-Management-Tool, das verwendet wird, um die aktive Box, deren Inhalte sowie die Sichtbarkeit der Untercontainer zu speichern und zu verwalten. Dadurch wird das Verhalten der interaktiven Diagramm-Boxen effizient gesteuert.

**Anpassbare Inhalte:**

* **activeBox** (number): Die aktuell ausgewählte Box im Diagramm. Dieser Zustand wird verwendet, um den Inhalt der aktivierten Box darzustellen.
* **activeBoxText** (number | null): Der aktuell ausgewählte Textcontainer innerhalb einer Box. Die Boxen können mehrere Textcontainer enthalten, die bei Aktivierung detaillierte Informationen anzeigen.
* **visibleContainers** (object): Enthält Informationen darüber, welche Boxen aktuell geöffnet und sichtbar sind.
* boxContents (Array): Enthält die Struktur und die Inhalte aller Boxen im Diagramm. Jede Box hat eine boxId, einen title, und eine Liste von items, die Unterinformationen zu den Boxen enthalten.

**Interaktive Elemente:**

* **Boxen**: Der Nutzer kann auf jede Box klicken, um mehr Informationen über die technischen Merkmale, Erlebniselemente, Lernprozesse oder Lernergebnisse zu erhalten.
* **Textcontainer innerhalb der Boxen**: Jede Box kann weitere Untercontainer mit detaillierten Informationen enthalten. Durch das Klicken auf diese Container werden zusätzliche Informationen angezeigt.
* **Pfeile**: Zusätzlich zu den Boxen gibt es Pfeilsymbole, die verwendet werden, um den Inhalt ein- und auszublenden. Dies sorgt für eine übersichtliche und interaktive Darstellung.

**Inhaltliche Anpassung:**

Die Inhalte der **KerresDiagramm**-Komponente können im Zustand-Store (useKerresStore) angepasst werden. Dieser Store speichert alle Boxen und deren Inhalte, welche dynamisch durch die Interaktion des Nutzers angezeigt werden.

## PublikationenPage

Die Publikationen-Seite ist eine zentrale Anlaufstelle für die neuesten Forschungsarbeiten und Publikationen des LEVIKO-XR-Projekts. Sie präsentiert eine Übersicht der wissenschaftlichen Ergebnisse und bietet Filtermöglichkeiten, um gezielt nach bestimmten Themen oder Beiträgen zu suchen.

**Verwendete Hauptkomponenten:**

* [HeroPublikationen](#HeroPublikationen)
* [Publikation](#Publikation)
* [ParallaxSection](#ParallaxSection)
* [MarqueeNews](#MarqueeNews)
* [Headline](#Headline)

### HeroPublikationen

**Pfad:**  
/Components/Hero/HeroPublikationen.tsx

**Styled-Component Pfad:**  
/Components/UI/Hero/HeroPublikationen.styled.tsx

**Store Pfad:**

/Components/stores/usePublikationenStore.tsx

**Beschreibung:**  
Die **HeroPublikationen**-Komponente bildet die heroische Einführungssektion der Publikationsseite. Sie stellt eine interaktive Suchleiste und vorgeschlagenen Tags vor, die die Benutzer zu relevanten Inhalten führen.

**Verwendete Bibliotheken:**

* Zustand: Wird als Zustand-Management-Tool verwendet, um den aktuellen Suchwert (searchValue) und den ausgewählten Tag (selectedTag) zu speichern und zu verwalten. Der Zustand sorgt dafür, dass die Suche und Filterung der Publikationen dynamisch erfolgt.

**Anpassbare Props:**

* **headline (string)**: Die Hauptüberschrift der Hero-Sektion. Sie fasst die Seite kurz zusammen und zieht die Aufmerksamkeit der Benutzer auf sich.
* **description (string)**: Eine detaillierte Beschreibung, die das Thema der Publikationen erklärt und den Benutzer dazu einlädt, die verschiedenen Beiträge zu entdecken.
* **allTags (string[]):** Ein Array aller verfügbaren Tags. Diese werden unter der Suchleiste angezeigt und helfen dem Benutzer, relevante Inhalte schnell zu finden.

**Funktionalitäten:**

* **searchValue** **(string):** Der aktuelle Suchbegriff, den der Benutzer eingibt. Dieser wird im Zustand-Store verwaltet und dazu verwendet, die angezeigten Tags und Publikationen zu filtern.
* **setSearchValue(value: string):** Setzt den aktuellen Suchwert und aktualisiert damit die Ergebnisse basierend auf dem eingegebenen Begriff.
* **setSelectedTag(tag: string):** Setzt den ausgewählten Tag und zeigt die entsprechenden Publikationen an.
* **filteredTags:** Basierend auf dem eingegebenen Suchbegriff werden nur die Tags angezeigt, die den Suchbegriff enthalten.

**Interaktive Elemente:**

* **Suchleiste:** Der Benutzer kann hier einen Suchbegriff eingeben, um relevante Tags zu filtern. Die Ergebnisse werden dynamisch angepasst, während der Benutzer tippt.
* **Tag-Vorschläge:** Eine Liste von Tags, die basierend auf dem Suchbegriff oder den vorgeschlagenen Tags gefiltert wird. Ein Klick auf einen Tag setzt den Suchwert und den ausgewählten Tag.
* **Tag-Ergebnisse:** Falls keine passenden Tags gefunden werden, zeigt die Komponente eine entsprechende Nachricht an.

**Inhaltliche Anpassung:**  
Die Inhalte der **HeroPublikationen**-Komponente, wie Überschrift und Beschreibung, können durch die entsprechenden Props angepasst werden. Die Tags werden automatisch aus dem Store bezogen und basierend auf den Suchbegriffen oder ausgewählten Tags gefiltert.

### Publikation

**Pfad:**  
/Components/Publikationen/Publikation.tsx

**Styled-Component Pfad:**  
/Components/UI/Publikationen/Publikation.styled.tsx

**Daten Pfad:**  
/Components/data/publikationen.ts

**Beschreibung:**  
Die Publikation-Komponente dient zur Darstellung einzelner wissenschaftlicher Veröffentlichungen oder Präsentationen. Jede Publikation wird als Karte mit einem Bild, einem Titel, einer Beschreibung und optionalen Tags angezeigt. Die Komponente unterstützt außerdem das Herunterladen von PDF-Dateien über einen entsprechenden Button. Die Inhalte der Publikation werden dynamisch über die Datei **publikationData.ts** verwaltet, wo alle relevanten Informationen wie Datum, Überschrift, Tags und Bilder definiert werden.

**Anpassbare Inhalte:**

* **date (string):** Das Veröffentlichungsdatum der Publikation. Dieses Datum wird prominent auf der Karte angezeigt.
* **headline (string):** Die Hauptüberschrift der Publikation, die das Thema zusammenfasst.
* **subHeadline (string):** Eine kurze Unterüberschrift, die weitere Details zur Publikation bietet.
* **imgSrc (string):** Der Pfad zur Bilddatei, die auf der Publikationskarte angezeigt wird.
* **imgAlt (string):** Der alternative Text, der für das Bild verwendet wird.
* **$objectPosition (string, optional):** Die Positionierung des Bildes innerhalb des Containers.
* **linkTo (string):** Der Link zur PDF-Datei oder einer externen Seite, zu der der Benutzer durch den Button geleitet wird.
* **button (string):** Der Text, der auf dem Button angezeigt wird, z.B. „Poster herunterladen“.
* **$buttonVariant (optional):** Bestimmt das Design des Buttons. Es gibt zwei Varianten: „secondary“ und „tertiary“.
* **description (JSX.Element | string):** Eine detaillierte Beschreibung der Publikation. Diese wird als Text oder JSX-Element unterhalb der Überschrift angezeigt.
* **tags (string[]):** Eine Liste von Schlagwörtern oder Kategorien, die unter der Beschreibung angezeigt werden und zur Klassifizierung der Publikation dienen.
* **isEven (boolean):** Bestimmt, ob das Layout der Publikation für abwechselnde Reihenfolgen von Text und Bild angepasst wird.
* **isPdf (boolean, optional):** Legt fest, ob der Button als PDF-Download oder als Link gerendert wird. Standardmäßig ist dieser Wert auf true gesetzt.

**Interaktive Elemente:**

* **Bild und Text:** Das Bild und die Beschreibung der Publikation sind visuell prominent dargestellt und geben dem Benutzer einen schnellen Überblick über den Inhalt.
* **Tags:** Tags werden unter der Beschreibung angezeigt und geben zusätzliche Informationen zur Klassifizierung der Publikation. Sie können vom Benutzer visuell erfasst werden, um eine schnelle Kategorisierung zu ermöglichen.
* **Button (PDF/Link):** Der Button am Ende der Publikation kann entweder als PDF-Download-Button oder als einfacher Link-Button gerendert werden. Der PDF-Button öffnet ein neues Fenster oder lädt die verlinkte Datei herunter, während der normale Button den Benutzer zu einer externen URL führt.

**Inhaltliche Anpassung:**  
Die Inhalte der Publikation-Komponente werden in der Datei **publikationData.ts** verwaltet. Diese Datei enthält ein Array von Publikationen, das die folgenden Informationen enthält:

* **date (string):** Das Veröffentlichungsdatum der Publikation.
* **headline (string):** Der Titel der Publikation.
* **subHeadline (string):** Eine kurze Zusammenfassung der Publikation.
* **tags (string[]):** Eine Liste von Tags, die zur Beschreibung der Publikation verwendet werden.
* **searchTags (optional, string[]):** Eine Liste von zusätzlichen Such-Tags, die für die Filterfunktion auf der Webseite verwendet werden können.
* **imgSrc (string):** Der Pfad zum Bild, das auf der Publikationskarte angezeigt wird.
* **imgAlt (string):** Der alternative Text für das Bild.
* **linkTo (string):** Der Link, der zu einer PDF-Datei oder einer externen Seite führt.
* **button (string):** Der Text, der auf dem Button angezeigt wird.
* **description (JSX.Element):** Eine detaillierte Beschreibung der Publikation, die als JSX-Element formatiert ist.

Durch Anpassung der Einträge in der **publikationData.ts**-Datei können leicht neue Publikationen hinzugefügt oder bestehende Publikationen geändert werden. Es ist auch möglich, zusätzliche Tags oder Beschreibungen hinzuzufügen, um den Benutzer umfassend über die Publikation zu informieren.

## TeamPage

Die TeamPage bietet eine umfassende Vorstellung des Teams hinter dem LEVIKO XR-Projekt. Sie präsentiert die Professoren und wissenschaftlichen MitarbeiterInnen der beteiligten Universitäten und Hochschulen und gibt Einblicke in die Institutionen, die an diesem innovativen Bildungsprojekt beteiligt sind. Die Seite ist in verschiedene Abschnitte unterteilt, die sowohl die Universitäten als auch die einzelnen Teammitglieder vorstellen. Durch interaktive Accordion-Komponenten und visuelle Elemente wird der Benutzer dazu eingeladen, mehr über die Expertise und das Engagement des Teams zu erfahren.

**Verwendete Hauptkomponenten:**

* [HeroTeam](#HeroTeam)
* [TeamAccordion](#TeamAccordion)
* [BildText](#BildText)
* [Headline](#Headline)

### HeroTeam

**Pfad:**  
/Components/Hero/HeroTeam.tsx

**Styled-Component Pfad:**  
/Components/UI/Hero/HeroTeam.styled.tsx

**Beschreibung:**  
Die HeroTeam-Komponente dient als visuelles Highlight für die Team-Seite und stellt ein zentrales Bild mit einer einladenden Überschrift bereit. Sie verwendet die Dark Mode-Funktionalität, um sich an den aktivierten Modus anzupassen. Die Komponente unterstützt das dynamische Einfügen eines Bildes sowie die Positionierung dieses Bildes im Container.

**Anpassbare Features:**

**Bild und Bildposition:**

* **$imgSrc:** Über diese Prop kann das Bild für den Hero-Bereich eingefügt werden.
* **$objectPosition:** Diese Prop ermöglicht es, die Position des Bildes im Container zu steuern (z. B. center, top, bottom). Standardmäßig ist sie auf center gesetzt.

**Überschrift:**

* **headline:** Die Hauptüberschrift wird dynamisch über diese Prop gesetzt. Sie passt sich ebenfalls dem Dark Mode an und bildet das zentrale Element der Komponente.

**Inhaltliche Anpassung:**

Die Inhalte der HeroTeam-Komponente können durch Anpassung der Props verändert werden:

* **$imgSrc:** Der Pfad zum Bild, das als Hintergrundbild angezeigt wird.
* **$objectPosition:** Die Bildposition innerhalb des Containers.
* **headline:** Die Überschrift, die im Vordergrund des Bildes dargestellt wird.
* **Dark Mode:** Die Farbanpassungen der Schrift und des Containers werden automatisch auf Grundlage des aktivierten Dark Modes gesteuert.

### TeamAccordion

**Pfad:**  
/Components/Accordion/TeamAccordion.tsx

**Styled-Component Pfad:**  
/Components/UI/Accordion/TeamAccordion.styled.tsx

**Daten Pfad:**  
/Components/data/TeamProfData.tsx

/Components/data/TeamDusData.tsx

/Components/data/TeamOsnaData.tsx

/Components/data/TeamTrosData.tsx

**Beschreibung:**  
Die TeamAccordion-Komponente dient zur Darstellung von Professoren und wissenschaftlichen Mitarbeitern im LEVIKO-XR-Projekt. Sie stellt detaillierte Informationen in einem Accordion-Layout dar, das durch Klicken geöffnet oder geschlossen werden kann. Jedes Teammitglied wird durch ein Bild, Logos, eine kurze Beschreibung und einen Button für zusätzliche Aktionen, wie z. B. das Senden einer E-Mail, dargestellt. Die Inhalte der TeamAccordion-Komponente werden dynamisch über zb. die Datei **TeamProfData.tsx** verwaltet, in der die relevanten Informationen, wie Name, Titel, Beschreibung, Bilder und Kontaktlinks der Teammitglieder, hinterlegt sind.

**Anpassbare Inhalte:**

* **id (string):** Eine eindeutige ID für jedes Teammitglied, die zur Identifikation innerhalb der Komponente verwendet wird.
* **title (JSX.Element | string):** Der Titel des Teammitglieds, der den Namen und die Position (z.B. „Verbundkoordination“ oder „Verbundpartner“) enthält. Dieser Titel wird als Überschrift im Accordion dargestellt.
* **description (string):** Eine kurze Beschreibung der Institution, der das Teammitglied angehört.
* **logoOne, logoTwo (string):** Pfade zu den Logos der jeweiligen Universitäten oder Einrichtungen, die das Teammitglied repräsentieren. In der Komponente werden diese Logos neben der Beschreibung angezeigt.
* **altLogoOne, altLogoTwo (string):** Alternativtexte für die Logos, um die Barrierefreiheit zu gewährleisten.
* **content (JSX.Element | string):** Detaillierte Informationen über das Teammitglied, die im geöffneten Zustand des Accordions angezeigt werden. Dieser Inhalt kann in Form von Text oder JSX-Elementen vorliegen.
* **imgSrc (string):** Der Pfad zum Bild des Teammitglieds, das im Accordion angezeigt wird.
* **imgAlt (string):** Alternativer Text für das Bild, um die Barrierefreiheit sicherzustellen.
* **$objectPosition (string, optional):** Die Positionierung des Bildes innerhalb des Containers. Optional, mit einem Standardwert von „center“.
* **linkTo (string):** Der Link, der bei Klick auf den Button aufgerufen wird, z.B. die E-Mail-Adresse des Teammitglieds.
* **button (string):** Der Text auf dem Button, z.B. „E-Mail“, um eine Aktion wie das Senden einer E-Mail auszuführen.
* **$buttonVariant (optional):** Stellt das Design des Buttons dar. Es gibt die Varianten „secondary“ und „tertiary“.

**Interaktive Elemente:**

* **Accordion-Elemente:** Jedes Teammitglied wird in einem Accordion dargestellt, das durch Klicken geöffnet und geschlossen werden kann. Im geöffneten Zustand werden weitere Informationen und Bilder sichtbar.
* **Buttons:** Ein Button am unteren Rand jedes geöffneten Accordion-Elements ermöglicht es, Aktionen wie das Senden einer E-Mail oder den Besuch einer externen Website auszuführen.

**Inhaltliche Anpassung:**

Die Inhalte der TeamAccordion-Komponente werden über die Datei TeamProfData.ts verwaltet. Diese Datei enthält ein Array von Teammitgliedern, das die folgenden Informationen enthält:

* **id (string):** Eine eindeutige Identifikationsnummer für jedes Teammitglied.
* title (JSX.Element | string): Der Titel, der den Namen und die Position des Teammitglieds enthält.
* **description (string):** Eine Beschreibung der Universität oder Institution des Teammitglieds.
* **logoOne, logoTwo (string):** Die Pfade zu den Logos der Institutionen.
* **altLogoOne, altLogoTwo (string):** Alternativtexte für die Logos.
* **content (JSX.Element | string):** Detaillierte Informationen zum Teammitglied, die beim Öffnen des Accordions angezeigt werden.
* **imgSrc (string):** Der Pfad zum Bild des Teammitglieds.
* **imgAlt (string):** Der alternative Text für das Bild.
* **linkTo (string):** Der Link zur E-Mail-Adresse oder einer anderen Aktion.
* **button (string):** Der Text auf dem Button.

Durch Änderungen in zb. der Datei **TeamProfData.tsx** können neue Teammitglieder hinzugefügt oder bestehende Informationen leicht angepasst werden.