**Projektentwicklung**

**Hauptkomponenten und ihre Methoden**

**DynamicComponentDirective**

**Beschreibung**: Diese Direktive wird verwendet, um dynamische Komponenten in der Ansicht zu platzieren.

**Warum verwendet**: Diese Direktive erhöht die Flexibilität, indem sie es ermöglicht, UI-Komponenten zur Laufzeit zu laden und zu rendern.

**Wichtige Methode**:

* constructor(viewContainerRef: ViewContainerRef): Initialisiert die Direktive und speichert eine Referenz auf den Container, in dem die dynamischen Komponenten platziert werden. Diese Methode wird aufgerufen, wenn die Direktive instanziiert wird, und stellt sicher, dass der viewContainerRef bereit ist, Komponenten zu hosten.

**ComponentFactoryService**

**Beschreibung**: Ein Service, der für die Erstellung und Verwaltung von UI-Komponenten zuständig ist.

**Warum verwendet**: Um die Erstellung und Verwaltung von dynamischen UI-Komponenten zentral zu steuern und sicherzustellen, dass die richtige Komponente basierend auf dem Typ geladen wird.

**Wichtige Methode**:

* getComponentFactory(type: string): Liefert die Factory für die angeforderte Komponente. Diese Methode sucht in einer Map nach der passenden Komponente und gibt die entsprechende ComponentFactory zurück, die für die Erstellung der Komponente verwendet wird.

**CodeGeneratorService**

**Beschreibung**: Dieser Service ist für die Verwaltung und Generierung des Codes basierend auf den ausgewählten und konfigurierten UI-Komponenten verantwortlich.

**Warum verwendet**: Um den generierten Code zu speichern, zu laden und bei Bedarf dynamisch zu erstellen, was die Effizienz und Wiederverwendbarkeit erhöht.

**Wichtige Methoden**:

* generateFormHtml(): Generiert das HTML-Formular basierend auf den ausgewählten Komponenten. Diese Methode durchläuft die Liste der Elemente und erstellt HTML-Code für jedes Element, der schließlich zu einem vollständigen Formular zusammengesetzt wird.
* generateTsCode(): Generiert den TypeScript-Code für die definierten Komponenten. Diese Methode erstellt den TypeScript-Code, der benötigt wird, um die konfigurierten Komponenten zu initialisieren und ihre Eigenschaften zu setzen.
* addElementToView(type: string, key?: string, width?: string, size?: string): Fügt ein neues Element zur Ansicht hinzu. Diese Methode erstellt ein neues UI-Element mit den angegebenen Eigenschaften und fügt es der internen Liste der Elemente hinzu, wodurch es in der Benutzeroberfläche angezeigt wird.
* removeItem(key: string): Entfernt ein Element aus der Ansicht. Diese Methode sucht nach dem Element mit dem angegebenen Schlüssel und entfernt es aus der internen Liste sowie aus der Benutzeroberfläche.
* moveItem(sourceIndex: number, targetIndex: number): Verschiebt ein Element innerhalb der Ansicht. Diese Methode ändert die Position eines Elements in der internen Liste, um seine Darstellung in der Benutzeroberfläche zu aktualisieren.
* changeElementPosition(key: string, newPosition: number): Ändert die Position eines Elements in der Ansicht. Diese Methode passt die Position eines Elements an, indem sie es an die neue Position in der internen Liste verschiebt.

**MainViewComponent**

**Beschreibung**: Diese Komponente ist das Hauptansichtsmodul der Anwendung. Sie beinhaltet die Logik zur Darstellung der Benutzeroberfläche und zur Handhabung der Benutzerinteraktionen.

**Warum verwendet**: Um die zentrale Logik und Darstellung der Hauptbenutzeroberfläche zu verwalten und eine interaktive und dynamische Benutzererfahrung zu ermöglichen.

**Wichtige Methoden**:

* renderElements(): Rendert die dynamischen Komponenten basierend auf der Konfiguration. Diese Methode erstellt und platziert die UI-Komponenten in der Ansicht, basierend auf der aktuellen Konfiguration, und stellt sicher, dass alle Komponenten korrekt angezeigt werden.
* handleDragStart(event: DragEvent): Wird aufgerufen, wenn ein Drag-Vorgang gestartet wird. Diese Methode initialisiert den Drag-Vorgang und speichert die notwendigen Informationen, um die Komponente später verschieben zu können.
* handleDragOver(event: DragEvent): Wird aufgerufen, während ein Drag-Vorgang über einem Drop-Ziel schwebt. Diese Methode verhindert das Standardverhalten und ermöglicht es, das Drag-Event zu akzeptieren.
* handleDrop(event: DragEvent): Wird aufgerufen, wenn ein Element fallen gelassen wird. Diese Methode verarbeitet das Drop-Event, aktualisiert die Position der Komponente und rendert die Ansicht neu.
* handleDragEnd(event: DragEvent): Wird aufgerufen, wenn ein Drag-Vorgang endet. Diese Methode bereinigt nach dem Drag-Vorgang und stellt sicher, dass die Benutzeroberfläche in einem konsistenten Zustand bleibt.

**CodeViewPopupComponent**

**Beschreibung**: Diese Komponente zeigt das generierte HTML- und TypeScript-Code im Popup-Fenster an.

**Warum verwendet**: Um den generierten Code anzuzeigen und zur Verfügung zu stellen, was den Entwicklern eine einfache Überprüfung und Bearbeitung ermöglicht.

**Wichtige Methoden**:

* open(): Öffnet das Popup. Diese Methode setzt das Popup sichtbar und ermöglicht die Interaktion mit dem angezeigten Code.
* close(): Schließt das Popup. Diese Methode verbirgt das Popup und verhindert weitere Interaktionen, bis es erneut geöffnet wird.
* copyHTMLCode(): Kopiert den generierten HTML-Code in die Zwischenablage. Diese Methode erleichtert das Kopieren und Einfügen des HTML-Codes an anderer Stelle.
* copyTsCode(): Kopiert den generierten TypeScript-Code in die Zwischenablage. Diese Methode erleichtert das Kopieren und Einfügen des TypeScript-Codes an anderer Stelle.

**AddElementComponent**

**Beschreibung**: Diese Komponente ermöglicht das Hinzufügen von neuen UI-Elementen.

**Warum verwendet**: Um Benutzern die Möglichkeit zu geben, neue UI-Komponenten hinzuzufügen und zu konfigurieren, was die Anpassbarkeit und Erweiterbarkeit der Anwendung erhöht.

**Wichtige Methoden**:

* addElement(): Fügt ein neues UI-Element hinzu. Diese Methode nimmt die Benutzereingaben entgegen, erstellt ein neues Element und fügt es der internen Liste der Elemente hinzu, sodass es in der Benutzeroberfläche angezeigt wird.

**ElementMenuComponent**

**Beschreibung**: Diese Komponente zeigt das Menü der verfügbaren UI-Komponenten an.

**Warum verwendet**: Um eine einfache Auswahl und Einfügung von UI-Komponenten zu ermöglichen und die Benutzerfreundlichkeit zu erhöhen.

**Wichtige Methoden**:

* selectComponent(componentType: string): Wählt eine Komponente aus dem Menü aus. Diese Methode wird verwendet, um die gewünschte Komponente zu selektieren und in die Ansicht einzufügen.

**KonfigurationPuppeComponent**

**Beschreibung**: Diese Komponente ermöglicht die Konfiguration der UI-Komponenten.

**Warum verwendet**: Um Benutzern die Möglichkeit zu geben, die Eigenschaften der UI-Komponenten zu ändern und anzupassen, was die Flexibilität und Anpassbarkeit der Anwendung erhöht.

**Wichtige Methoden**:

* configureComponent(componentId: string, config: any): Konfiguriert die ausgewählte Komponente. Diese Methode ermöglicht es dem Benutzer, die Eigenschaften der ausgewählten Komponente zu ändern und die Änderungen zu speichern.

**Herausforderungen und Lösungen**

**Dynamisches Laden und Konfigurieren von UI-Komponenten zur Laufzeit**

**Herausforderung**: Dynamisches Laden und Konfigurieren von UI-Komponenten basierend auf Benutzerinteraktionen.

**Lösung**: Verwendung der DynamicComponentDirective und des ComponentFactoryService zur Erhöhung der Flexibilität und zentralen Steuerung.

**Implementierung von Drag-and-Drop-Funktionalitäten für UI-Komponenten**

**Herausforderung**: Verbesserung der Benutzerfreundlichkeit durch Drag-and-Drop-Funktionalitäten.

**Lösung**: Implementierung von Methoden wie handleDragStart(), handleDragOver(), handleDrop(), handleDragEnd() in der MainViewComponent.

**Speichern und Laden von Konfigurationen der UI-Komponenten**

**Herausforderung**: Effizientes Speichern und Laden von Konfigurationen der UI-Komponenten.

**Lösung**: Implementierung von Methoden im CodeGeneratorService zum Speichern und Laden der Konfigurationen im Local Storage.

**Zeitplanung**

**Gantt-Diagramm**

Hier ist eine exemplarische Zeitplanung für das Projekt:

| **Aufgabe** | **Dauer (Stunden)** | **Startdatum** | **Enddatum** |
| --- | --- | --- | --- |
| Analysephase | 5 | 01.05.2024 | 02.05.2024 |
| Planungsphase | 8 | 03.05.2024 | 05.05.2024 |
| Entwicklungsphase | 40 | 06.05.2024 | 15.05.2024 |
| Testphase | 10 | 16.05.2024 | 18.05.2024 |
| Fehleranalyse und Behebung | 7 | 19.05.2024 | 21.05.2024 |
| Abnahme durch Abteilungsleiter | 2 | 22.05.2024 | 22.05.2024 |
| Projektdokumentation | 8 | 23.05.2024 | 25.05.2024 |