**Dokumentation zur betrieblichen Projektarbeit**

**IHK Abschlussprüfung**

**Projekteinführung**

**Inhaltsverzeichnis**

1. [**Einleitung**](#einleitung)
2. [**Projektumfeld**](#projektumfeld)
3. [**Projektumsetzung**](#projektumsetzung)
4. [**Projektziel**](#projektziel)
5. [**Projektanalyse**](#projektanalyse)
   * **5.1** [**Ist-Analyse**](#ist-analyse)
   * **5.2** [**Soll-Analyse**](#soll-analyse)
   * **5.3** [**Projektbegründung**](#projektbegr%C3%BCndung)
   * **5.4** [**"Make-or-Buy"-Entscheidung**](#make-or-buy-entscheidung)
6. [**Projektplanung**](#projektplanung)
   * **6.1** [**Projektphasen**](#projektphasen)
   * **6.2** [**Ressourcenplanung**](#ressourcenplanung)
   * **6.3** [**Zielplattform**](#zielplattform)
   * **6.4** [**Architekturdesign**](#architekturdesign)
7. [**Projektentwicklung**](#projektentwicklung)
   * **7.1** [**Hauptkomponenten und ihre Methoden**](#hauptkomponenten-und-ihre-methoden)
     + **7.1.1** [**DynamicComponentDirective**](#dynamiccomponentdirective)
     + **7.1.2** [**ComponentFactoryService**](#componentfactoryservice)
     + **7.1.3** [**CodeGeneratorService**](#codegeneratorservice)
     + **7.1.4** [**MainViewComponent**](#mainviewcomponent)
     + **7.1.5** [**CodeViewPopupComponent**](#codeviewpopupcomponent)
     + **7.1.6** [**AddElementComponent**](#addelementcomponent)
     + **7.1.7** [**ElementMenuComponent**](#elementmenucomponent)
     + **7.1.8** [**KonfigurationPuppeComponent**](#konfigurationpuppecomponent)
8. [**Herausforderungen und Lösungen**](#herausforderungen-und-l%C3%B6sungen)
9. [**Zeitplanung**](#zeitplanung)
   * **9.1** [**Gantt-Diagramm**](#gantt-diagramm)
10. [**Fazit**](#fazit)
11. [**Anhang**](#anhang)

**1. Einleitung**

**Diese Dokumentation beschreibt die Entwicklung eines Code-Generators für Dialoge in einer Angular- und TypeScript-Anwendung. Das Projekt wurde im Rahmen der Abschlussprüfung für Fachinformatiker Anwendungsentwicklung durchgeführt und umfasst die Implementierung eines Tools, das verschiedene UI-Komponenten in Dialogfenster einfügt und konfiguriert.**

**Ziel des Projekts ist es, eine benutzerfreundliche und flexible Anwendung zu entwickeln, die es ermöglicht, Dialogfenster dynamisch zu erstellen und anzupassen. Durch die Verwendung moderner Webtechnologien wie Angular und TypeScript soll eine leistungsfähige und wartbare Lösung bereitgestellt werden. Der Code-Generator soll dabei helfen, die Effizienz bei der Entwicklung von Benutzeroberflächen zu steigern und die Fehleranfälligkeit zu reduzieren.**

**Im Rahmen des Projekts wurden mehrere Schlüsselkomponenten entwickelt, darunter eine Direktive zur dynamischen Platzierung von Komponenten, Services zur Verwaltung und Generierung von Code sowie eine benutzerfreundliche Oberfläche zur Konfiguration der Dialoge. Die Entwicklung erfolgte in mehreren Phasen, von der Analyse über die Planung und Implementierung bis hin zur abschließenden Testung und Dokumentation.**

**Diese Dokumentation gibt einen umfassenden Überblick über die Projektumsetzung, erläutert die verwendeten Technologien und Architekturen und beschreibt die einzelnen Komponenten und ihre Funktionalitäten im Detail. Darüber hinaus werden die Herausforderungen und Lösungen, die während der Entwicklung auftraten, dargestellt und eine Zeitplanung sowie ein Fazit gegeben.**

**2.1 Projektumfeld**

Das Projekt wurde im Rahmen der Abschlussprüfung zum Fachinformatiker für Anwendungsentwicklung entwickelt. Ziel ist die Erstellung einer Benutzeroberfläche in Angular und TypeScript, die als Code-Generator für Dialoge dient. Die Anwendung ermöglicht es, verschiedene UI-Komponenten aus einem Menü in ein Dialogfenster einzufügen und zu konfigurieren.

**2.2 Projektumsetzung**

Das Projekt wurde innerhalb von 80 Stunden realisiert, einschließlich der Implementierung, Tests und der Erstellung einer umfassenden Dokumentation. Es handelt sich um ein internes Projekt, das im Rahmen der Abschlussprüfung realisiert wurde.

**2.3 Projektziel**

Ziel des Projekts ist die Bereitstellung einer Anwendung, die die Erstellung und Verwaltung von Dialogfenstern durch eine einfache Drag-and-Drop-Funktionalität und eine konfigurierbare Benutzeroberfläche erleichtert.

**Projektanalyse**

**3.1 Ist-Analyse**

Die bestehende Lösung für die Erstellung von Dialogfenstern war ineffizient und zeitaufwendig. Manuelle Konfigurationen und fehlende Automatisierungen führten zu hoher Fehleranfälligkeit und unübersichtlichem Code.

**3.2 Soll-Analyse**

Die neue Lösung soll eine automatisierte, benutzerfreundliche Oberfläche bieten, die die Erstellung und Konfiguration von Dialogfenstern vereinfacht. Die Anwendung soll flexible, wiederverwendbare Komponenten bieten, die dynamisch geladen und konfiguriert werden können.

**3.3 Projektbegründung**

Durch die Implementierung der neuen Lösung werden Zeit und Ressourcen gespart, die Benutzerfreundlichkeit erhöht und die Fehleranfälligkeit reduziert. Eine automatisierte Lösung verbessert die Effizienz und Konsistenz bei der Erstellung von Dialogfenstern.

**2.4 "Make-or-Buy"-Entscheidung**

Nach Prüfung bestehender Lösungen wurde entschieden, eine individuelle Lösung zu entwickeln, um spezifische Anforderungen und Anpassungen optimal zu erfüllen.

**Projektplanung**

**4.1 Projektphasen**

* Analysephase
* Planungsphase / Entwurfsphase
* Entwicklungsphase
* Testphase
* Fehleranalyse
* Abnahme durch Abteilungsleiter
* Projektdokumentation

**4.2 Ressourcenplanung**

Die Entwicklung fand auf einem leistungsfähigen Entwicklungsrechner mit den notwendigen Softwaretools statt. Es entstanden keine zusätzlichen Kosten, da alle verwendeten Tools und Frameworks bereits lizenziert waren oder kostenlos zur Verfügung standen.

**4.3 Zielplattform**

Die Zielplattform ist eine Angular-Anwendung, die in TypeScript entwickelt wurde. Sie integriert sich nahtlos in bestehende Systeme und nutzt moderne Webtechnologien.

**4.4 Architekturdesign**

Das Projekt folgt dem MVC-Architekturprinzip (Model-View-Controller), um eine klare Trennung von Datenmodell, Benutzeroberfläche und Anwendungslogik zu gewährleisten. Dies fördert die Wartbarkeit und Erweiterbarkeit der Anwendung.

**Projektentwicklung**

**Hauptkomponenten und ihre Methoden**

**DynamicComponentDirective**

**Beschreibung**: Ermöglicht das dynamische Platzieren von Komponenten in der Ansicht.

**Warum verwendet**: Erhöht die Flexibilität, UI-Komponenten zur Laufzeit zu laden und zu rendern.

**Wichtige Methode**:

* constructor(viewContainerRef: ViewContainerRef): Initialisiert die Direktive und speichert eine Referenz auf den Container.

**ComponentFactoryService**

**Beschreibung**: Service zur Erstellung und Verwaltung von UI-Komponenten.

**Warum verwendet**: Zentrale Steuerung der Erstellung und Verwaltung dynamischer UI-Komponenten.

**Wichtige Methode**:

* getComponentFactory(type: string): Liefert die Factory für die angeforderte Komponente. Diese Methode wird verwendet, um die entsprechende ComponentFactory basierend auf dem übergebenen Typ zu erhalten.

**CodeGeneratorService**

**Beschreibung**: Verwalten und Generieren von Code basierend auf konfigurierten UI-Komponenten.

**Warum verwendet**: Dynamisches Erstellen, Speichern und Laden des generierten Codes.

**Wichtige Methoden**:

* generateFormHtml(): Generiert das HTML-Formular basierend auf den ausgewählten Komponenten. Diese Methode durchläuft die Liste der Komponenten und erstellt das entsprechende HTML.
* generateTsCode(): Generiert den TypeScript-Code für die definierten Komponenten. Diese Methode erstellt den notwendigen TypeScript-Code basierend auf den konfigurierten Komponenten.
* 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.
* removeItem(key: string): Entfernt ein Element aus der Ansicht. Diese Methode löscht ein Element basierend auf seinem Schlüssel aus der internen Liste.
* moveItem(sourceIndex: number, targetIndex: number): Verschiebt ein Element innerhalb der Ansicht. Diese Methode ändert die Position eines Elements in der internen Liste basierend auf den angegebenen Indizes.
* changeElementPosition(key: string, newPosition: number): Ändert die Position eines Elements in der Ansicht. Diese Methode verschiebt ein Element an eine neue Position innerhalb der internen Liste basierend auf dem übergebenen Schlüssel und der neuen Position.

**MainViewComponent**

**Beschreibung**: Hauptansichtsmodul der Anwendung, verwaltet die Benutzeroberfläche und Benutzerinteraktionen.

**Warum verwendet**: Verwaltung der zentralen Logik und Darstellung der Hauptbenutzeroberfläche.

**Wichtige Methoden**:

* renderElements(): Rendert die dynamischen Komponenten basierend auf der Konfiguration. Diese Methode durchläuft die Liste der konfigurierten Elemente und erstellt für jedes Element die entsprechende Komponente.
* handleDragStart(event: DragEvent): Handhabt den Beginn einer Drag-and-Drop-Operation. Diese Methode wird aufgerufen, wenn der Benutzer eine Komponente zu ziehen beginnt.
* handleDragOver(event: DragEvent): Erlaubt das Ablegen eines Elements. Diese Methode wird aufgerufen, wenn eine Komponente über einen gültigen Drop-Bereich gezogen wird.
* handleDrop(event: DragEvent): Handhabt das Ablegen eines Elements. Diese Methode wird aufgerufen, wenn eine Komponente in einem gültigen Drop-Bereich abgelegt wird.
* handleDragEnd(event: DragEvent): Handhabt das Ende einer Drag-and-Drop-Operation. Diese Methode wird aufgerufen, wenn der Benutzer das Ziehen einer Komponente beendet.

**CodeViewPopupComponent**

**Beschreibung**: Zeigt generierten HTML- und TypeScript-Code im Popup-Fenster an.

**Warum verwendet**: Ermöglicht die Anzeige und Bearbeitung des generierten Codes.

**Wichtige Methoden**:

* open(): Öffnet das Popup.
* close(): Schließt das Popup.
* copyHTMLCode(): Kopiert den generierten HTML-Code in die Zwischenablage. Diese Methode wird verwendet, um den angezeigten HTML-Code zu kopieren.
* copyTsCode(): Kopiert den generierten TypeScript-Code in die Zwischenablage. Diese Methode wird verwendet, um den angezeigten TypeScript-Code zu kopieren.

**AddElementComponent**

**Beschreibung**: Ermöglicht das Hinzufügen neuer UI-Elemente.

**Warum verwendet**: Benutzern die Möglichkeit geben, neue UI-Komponenten hinzuzufügen und zu konfigurieren.

**Wichtige Methoden**:

* addElement(type: string): Fügt ein neues UI-Element hinzu. Diese Methode nimmt die Benutzereingaben entgegen und erstellt basierend darauf ein neues Element, das der Liste der Elemente hinzugefügt wird.

**ElementMenuComponent**

**Beschreibung**: Zeigt das Menü der verfügbaren UI-Komponenten an.

**Warum verwendet**: Einfache Auswahl und Einfügung von UI-Komponenten.

**Wichtige Methoden**:

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

**KonfigurationPuppeComponent**

**Beschreibung**: Ermöglicht die Konfiguration der UI-Komponenten.

**Warum verwendet**: Benutzern die Möglichkeit geben, die Eigenschaften der UI-Komponenten zu ändern und anzupassen.

**Wichtige Methoden**:

* configureComponent(key: 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 im Detail**

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

**Herausforderung**: Es war erforderlich, UI-Komponenten dynamisch zu laden und zu konfigurieren, basierend auf Benutzerinteraktionen.

**Lösung**: Die DynamicComponentDirective und der ComponentFactoryService wurden verwendet, um diese Flexibilität zu erreichen. Die DynamicComponentDirective ermöglicht es, Komponenten dynamisch in der Ansicht zu platzieren, während der ComponentFactoryService die Erstellung und Verwaltung dieser Komponenten zentral steuert.

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

**Herausforderung**: Die Benutzerfreundlichkeit sollte durch Drag-and-Drop-Funktionalitäten verbessert werden, um UI-Komponenten einfach zu verschieben und neu anzuordnen.

**Lösung**: Methoden wie handleDragStart(), handleDragOver(), handleDrop(), handleDragEnd() wurden in der MainViewComponent implementiert, um Drag-and-Drop-Operationen zu unterstützen. Diese Methoden handhaben die verschiedenen Phasen des Drag-and-Drop-Prozesses und aktualisieren die UI entsprechend.

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

**Herausforderung**: Die Konfigurationen der UI-Komponenten mussten gespeichert und bei Bedarf wieder geladen werden, um die Benutzerarbeit zu erleichtern.

**Lösung**: Im CodeGeneratorService wurden Methoden wie saveElementList() und loadElementList() implementiert, um die Konfigurationen im Local Storage zu speichern und zu laden. Dadurch können Benutzer ihre Arbeit speichern und später fortsetzen.

**Zeitplanung**

**Gantt-Diagramm**

Hier ist eine exemplarische Zeitplanung für das Projekt:

| **Aufgabe** | **Dauer (Stunden)** |
| --- | --- |
| Analysephase | 5 |
| Planungsphase | 8 |
| Entwicklungsphase | 40 |
| Testphase | 10 |
| Fehleranalyse und Behebung | 7 |
| Abnahme durch Abteilungsleiter | 2 |
| Projektdokumentation | 8 |

(Ein Gantt-Diagramm wird hier eingefügt, das die Projektphasen und den Zeitplan darstellt)

**Fazit**

In dieser Projektarbeit wurde ein Tool zur Erstellung und Konfiguration von UI-Komponenten in Angular und TypeScript entwickelt. Die Implementierung umfasste die Entwicklung von mehreren Komponenten und Services sowie die Erstellung einer umfassenden Dokumentation. Das Projekt wurde innerhalb des vorgegebenen Zeitrahmens erfolgreich abgeschlossen.

9.Anhang

* UML-Klassendiagramme