Ein Bild, das Text, Schrift, Logo, Grafiken enthält.

KI-generierte Inhalte können fehlerhaft sein.

Bachelorarbeit

Internationale Hochschule Fernstudium

Studiengang: B. Sc. Wirtschaftsinformatik

Die Vorteile der High-Code Entwicklung im Vergleich zur Low-Code Entwicklung

Evsin Rahmiev Matrikelnummer: 32105477

Akademiestr. 6

68159 Mannheim

Betreuungsperson: Prof. Dr. Thorsten Fröhlich 02.05.2025

Abstract

Im Bereich der Softwareentwicklung bietet sich neben der Möglichkeit, eine Software konventionell über High-Code Entwicklung mittlerweile auch die Option, über Low-Code zu entwickeln. Beide Ansätze haben unterschiedliche Vor- und Nachteile, je nachdem, was für Ziele hinter der Entwicklung stehen. Die vorliegende Arbeit hat die Aufgabe, zu beantworten, welche Vorteile die High-Code Entwicklung im Vergleich zu Low-Code anbietet. Zum Zweck der Arbeit und zur Beantwortung der Forschungsfrage wurde ein Case Study durchgeführt, bei der eine einfache Anwendung zum Deutschlernen einerseits über die etablierten Webentwicklungstechnologien HTML, CSS und JavaScript, also High-Code Werkzeuge, und andererseits über die Low-Code Entwicklungsanwendung *Zoho Creator* implementiert wurde. Es wurde untersucht, welche der beiden Entwicklungsansätze besser geeignet ist, um die konzipierte Lernanwendung mit ihren Funktionalitäten wie z. B. Verlinkung zwischen den Seiten der Anwendung, Manipulation der UI-Elemente, die grafische und funktionelle Anpassung von Formularen umzusetzen. Die Ergebnisse zeigen, dass sich die geplanten Anwendungskomponenten über High-Code Entwicklung näher an die Anforderungen implementieren lassen, weil diese Standardwebentwicklung mehr Anpassungsmöglichkeiten bietet. Über Low-Code hat man eher den Vorteil, dass man mit weniger Programmierkenntnissen eine Anwendung bauen kann, die dafür aber technisch weniger vielfältig ist.

Inhaltsverzeichnis

AbstractI

I. AbkürzungsverzeichnisII

II. AbbildungsverzeichnisIII

III. TabellenverzeichnisIV

1. Einleitung1

1.1 Forschungsziel4

1.2 Forschungsfragen4

1.3 Struktur der Arbeit4

2. Theoretische Fundierung1

3. Methodik/Forschungsdesign2

4. Forschungsergebnisse3

4.1 4

5. Interpretation12

6. Fazit15

IV. Literaturverzeichnis16

I. Abkürzungsverzeichnis

1. HC – High-Code1

3. LC/ NC – Low-Code / No-Code3

4. lDE – Integrated Development Environment2

5. HTML – Hypertext Markup Language1

6. CSS – Cascading Style Sheets1

7. DOM – Document Object Model1

8. OOP – Object Oriented Programming1

II. Abbildungsverzeichnis

Abb 1: Formel zur Berechnung des VI2

III. Tabellenverzeichnis

Tab: Projekt- und Releaseplan mit Iterationsraster, Phasen und Meilensteinen für Vorabnahmen, Releases und Abnahmen13

1. Einleitung

Die Technologie der Low-Code/No-Code Entwicklung bietet einen alternativen Ansatz zur Softwareentwicklung, mithilfe derer die Entwicklungszeit für eine ausgereifte Softwarelösung drastisch reduziert werden kann. Dies ist möglich dank der Art und Weise der LC/NC-Entwicklung, bei der Systemkomponenten und Module zum Teil durch das einfache Auswählen bzw. Drag-and-Drop von vorprogrammierten UI-Elementen wie z. B. Auswahlfelder, Tabellen oder Grafiken implementiert werden können. Dies stellt eine Erleichterung im Vergleich zu der bereits etablierten und mehr bekannten High-Code Entwicklung, bei der die Umsetzung ähnlicher Systemkomponenten in jedem Fall das Schreiben von Programmcode erfordert. Diese Unterschiede der beiden Programmieransätze lassen sich bei der Entwicklung von LexiCode, eine Anwendung zum Deutschlernen, auch bemerkbar. Die Anwendung, die grob beschrieben eine reine Frontend Anwendung ist und unter anderem Theorie zu den grammatikalischen Grundlagen der deutschen Sprachen übermitteln und diese durch interaktive Übungsaufgaben überprüfen soll. LexiCode kann in der Theorie sowohl über HC- als auch LC/NC-Entwicklung umgesetzt werden. Die Fragen, die in der Praxis allerdings beantwortet werden müssen, sind, welcher der beiden Ansätze ermöglicht, dass die konzipierten Systemmerkmale näher an die Anforderungen programmiert werden können und welcher der Ansätze bietet breiter Anpassungsmöglichkeiten für LexiCode an.

1.1 Forschungsziel

Die vorliegende Arbeit hat die Aufgabe, nachzuweisen, dass die klassische HC-Entwicklung besser geeignet ist, um die Anforderungen an LexiCode technisch umsetzen zu können. Es wird erforscht, welcher der beiden Entwicklungsansätze eine Umsetzung ermöglicht, die die Anwendung LexiCode möglichst interaktiv, also für Interaktionen zwischen Nutzern und der Anwendung sorgt, visuell und funktionell anpassungsfähig macht und eine dynamische Navigation zwischen den Seiten der Anwendung ermöglicht. Darüber hinaus wird geprüft, welche der beiden Ansätze bessere Anpassungsmöglichkeiten bietet, wenn die Anwendung für mobile Endgeräte erweitert wird.

Um dieses Ziel zu erreichen, werden die folgenden Forschungsfragen beantwortet:

**1. Können über HC- oder LC/NC-Entwicklung Hyperlinks angelegt werden, die die User auch vom Inhalt einer Seite, also nicht nur von der globalen Navigation heraus, auf einer anderen Seite innerhalb der Anwendung umleiten können?**

Anwendungen verfügen normalerweise über eine globale Navigation, die die Benutzer auf die Hauptseiten der Anwendung verleiten können. Oftmals werden Links zu anderen Seiten auch im Inhalt einer Seite eingebaut, damit nicht alle verwandten Inhalte auf der gleichen Seite stehen und somit eine Übersichtlichkeit erreicht wird.

**2. Über welchen der Entwicklungsansätze können die Seitenelemente effektiver oder überhaupt manipuliert werden, um mehr Nutzungseffekte zu erreichen?**

Bei der HC-Entwicklung ist in der Programmiersprache JavaScript das Konzept bzw. Werkzeug des DOMs bekannt, welches die Möglichkeit bietet, dass an bestehende Elemente der der Anwendung neue Elemente wie Überschriften, Zeilen oder Bilder angeknüpft oder bestehende Elemente gelöscht werden (Flanagan, 2020, S. 416). Eine Unterfrage zu dieser Forschungsfrage ist, ob die gleiche Manipulation bei LC/NC vorhanden ist.

**3. Ist die Auswahl an Anpassungsmöglichkeiten für Webformulare breiter bei HC- oder LC/NC-Anwendungen?**

Auf Webanwendungen sind häufig Formulare zu finden. Diese erfüllen die gleiche Aufgabe wie Formulare in Papierform, mit dem Unterschied, dass diese digitalen Formulare u.a. die abgefragten Daten schneller und direkt an die abfragende Instanz schicken können. Die Anpassungsmöglichkeiten sollen die optische Gestaltung der Formulare und ihre Funktionalität umfangen.

**4. Lassen sich sowohl über HC als auch über LC/NC inzeiligen Eingabefelder einbauen und wie gut lassen sich diese anpassen?**

Beim Sprachenlernen sind Übungsaufgaben üblich, bei denen ein Wort, mehrere Worte oder komplette Satzteile ausgelassen werden und stattdessen sich Eingabefelder auf deren Plätzen befinden. Solche Felder sind eine Anforderung an LexiCode und lassen die Frage stellen, ob sich diese über HC oder LC/NC einbauen lassen und wenn ja, ob sie angpeasst werden können wie z. B. in der Breite, in der Höhe, in der Randdichte oder die Hintergrundfarbe.

**5. Können Funktionen umgesetzt werden, die benutzerdefinierte Inhalte lokal und sessionunabhängig speichern können?**

Bei der Arbeit mit einer Anwendung werden oft Daten in der Anwendung angegeben, die nach einer kurzen oder sogar längeren Zeit wieder abrufbar sein sollten. Obwohl eine Datenbankanbindung eine gängige Lösung für Datenspeicherung ist, ist für die Speicherung von Daten aller Arten nicht zwingend erforderlich. Antworten zu Übungsaufgaben oder Notizen können z. B. auf dem Browser gespeichert werden und sind trotzdem wieder in der Anwendung zu finden.

6. Fazit

IV. Literaturverzeichnis

Baumgarten, C., Rainer, E. & Stich, S. (2024): Professionelle Softwareentwicklung mit Low Code optimieren – eine Fallstudie. In: *HMD Praxis der Wirtschaftsinformatik*. Springer Verlag.

Brandt-Pook, H. & Kollmeier, R. (2020): *Softwareentwicklung kompakt und verständlich. Wie Softwaresysteme entstehen*. (3. Auflage). Springer Verlag.

Böhler, T. (2023): Software einfacher, flexibler und leichtfüßiger entwickeln. In: *Produktion*. (12):16.

Elshan, E. & Binzer, B. (2024): Mehr als ein Trend?: Wie Low-Code die digitale Transformation unterstützt. In: *HMD Praxis der Wirtschaftsinformatik*. 1070-1087.

Flanagan, D. (2020): *JavaScript: The definitive guide: Master the world’s most-used programming language*. O’Reilly Media, Incorporated

Frank, B. (2023): *ERP:* The next big thing: Sind Lösungen auf Low-Code-Basis die Zukunft? In: *IT-Management*. 50-54.

Hensen, U. (2023): Maßgeschnittene Software mittels Low Code. In: *Factory Innovation*. 33-37.

Schreiner, K. (2024): Entwicklungszyklen halbieren mit Low Code. In: *Digital Engineering*. 44-45.

Siroker, D. & Koomen, P. (2013): *A/B Testing. The Most Powerful Way to Turn Clicks into Customers*. Wiley.

Spierling, D. (2023): Mehr IT-Nachhaltigkeit und Klimaschutz durch Low-Code Development. In: *Wirtschaftsinformatik & Management*. 107–113.

Eidesstattliche Erklärung

Ein Bild, das Text, Schrift, Logo, Grafiken enthält.

KI-generierte Inhalte können fehlerhaft sein.

Hiermit versichere ich an Eides statt, dass ich die Abschlussarbeit selbständig und ohne Inanspruchnahme fremder Hilfe angefertigt habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen. Ich erkläre mich damit einverstanden, dass die Arbeit mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate überprüft wird.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, Datum Unterschrift