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, die Frage 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 geplanten 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.

**Schlüsselwörter**: High-Code, Low-Code, Softwareentwicklung, Anwendung

Abstract

Apart from programming an application from scratch suing conventional high-code development, the field of software development also offers the opportunity of developing an application using low-code development. Both approaches have different advantages and disadvantages depending on the goals oft he development. The present project aims to answer the question of which advantages high-code development offers compared to low-code development. Fort he purpose oft he project and in ordert to answer the research question, a case study was conducted where a simple application for learning German was developed by using established web development technologies like HTML, CSS and JavaScript on one hand and the low-code development platfrom Zoho Creator on the other. The experiment aimed to discover which one oft he two development methods is more suitable for programming the conceived learning application and its planned features such as links between the application’s pages, manipulation of ist ui-elements and the graphical i.e. visual and functional customization of forms. The results show that development through high-code allows for the application’s requirements tob e met to a higher level because standard development offers more customization opportunities. Low-code development offers more of an advantage in terms of coding an application with less programming skills, which in terms makes the application less technically diverse.

**Key words**: high-code, low-code, software development, application

Inhaltsverzeichnis

AbstractI

I. AbkürzungsverzeichnisII

II. AbbildungsverzeichnisIII

III. TabellenverzeichnisIV

1. Einleitung1

1.1 Forschungsziel4

1.2 Forschungsfragen4

1.3 These4

1.4 Wert und Zielpublikum4

1.5 Anwendungsbereich und Einschränkungen4

1.6 Struktur des Dokuments4

2. Theoretische Fundierung1

2.1 Terminologie und Definitionen4

2.2 Relevante Theorien und Konzepte4

3. Methodik/Forschungsdesign2

4. Forschungsergebnisse3

5. Interpretation12

6. Fazit15

IV. Literaturverzeichnis16

I. Abkürzungsverzeichnis

1. HC – High-Code1

3. LC/ NC – Low-Code / No-Code3

4. LCAP – Low-Code-Applikationsplattform2

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: Iron Triangle2

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, also die geplanten Systemkomponenten im Sinne von UI-Elementen und Funktionalitäten möglichst nah an die geplanten Komponenten zu halten. 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, eine dynamische Navigation zwischen den Seiten der Anwendung ermöglicht, eine Manipulation der UI-Elemente anhand der Nutzeraktivitäten bereitstellt, den Einbau von inzeiligen Eingabefeldern erlaubt und sitzungsunabhängig Nutzerdaten im Browser speichern kann.. Darüber hinaus wird geprüft, welche der beiden Ansätze bessere Anpassungsmöglichkeiten bietet, wenn die Anwendung für mobile Endgeräte erweitert wird.

1.2 Forschungsfragen

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.

1.3 These

Die gestellten Forschungsfragen setzen Erwartungen an die Forschung sowie eine klare Vorstellung für die Endergebnisse. Unabhängig von diesen Erwartungen kann die Aussage formuliert werden, dass HC-Entwicklung aus der Sicht der breiten Verfügbarkeit von Werkzeugen zur Umsetzung der jeweiligen Anforderungen einen einfacheren Entwicklungsprozess zulässt.

1.4 Wert und Zielpublikum

Die Ergebnisse dieser Forschung leisten einen Beitrag zur Debatte über den effektiveren Entwicklungsansatz. Die verteidigte Ansicht, dass die LC/NC-Entwicklung die Zukunft der Softwareentwicklung wäre, beschränkt sich nur auf den Aspekt der Benutzerfreundlichkeit von LC/NC-Entwicklungsplattformen bzw. die Einfachheit, über solche Plattformen eine Software ins Leben zu rufen. Zwei Aspekte, die allerdings bei dieser Ansicht oft ignoriert werden, sind die Kompatibilität von LC/NC-Anwendungen mit anderen Systemen und die Verfügbarkeit von Werkzeugen, die weitere Konfigurations- und Anpassungsmöglichkeiten bieten oder auch für Wiederverwendbarkeit des entwickelten Programmcodes, Skalierbarkeit und Wartung ermöglichen. Dafür werden Bibliotheken, Frameworks und IDEs eingesetzt. Die Ergebnisse der Forschung sind für Softwareentwickler interessant und zwar unabhängig von der Erfahrung, Ausbildung, Branche oder Fachgebiet.

1.5 Anwendungsbereich und Einschränkungen

Die Relevanz der Forschungsergebnisse schränkt sich auf kleinere bis mittelgroßen Anwendungen ein, weil dies, zumindest im Rahmen dieser Forschung, die vorgesehene Größe der Anwendung LexiCode ist. Ferner sind die Ergebnisse für Personen interessant, die über wenig Erfahrung als Softwareentwickler verfügen, in eher kleinere Teams oder sogar allein die Entwicklungsarbeiten durchführen werden, mit weniger hardware- und softwaretechnischen Ressourcen arbeiten werden, eine überschaubare Anwendung entwickeln möchten und sich überlegen, ob sie sich für HC oder LC/NC entscheiden sollen. Die Ergebnisse der Forschung sind auch für Lernanwendungen relevant, die andere Inhalte als eine Fremdsprache übermitteln wollen.

Die Forschung befasst sich nicht mit Prozessen und Diensten, die die Speicherung oder das Abrufen von Anwendungs- oder Nutzerdaten von einem System oder Schnittstelle, die im Hintergrund der Anwendung laufen, wie z. B. eine Datenbank, ein Server oder eine Schnittstelle, weil die Umsetzung solcher Lösungen zusammen mit den in den vorherigen Kapiteln beschriebenen Funktionalitäten den vorgesehenen Rahmen dieser Forschung übersprengen würde. Insofern sind die Ergebnisse der Forschung eher für Frontend-Lösungen relevant.

1.6 Struktur des Dokuments

In dem darauffolgenden Teil des Dokuments werden die konkreten Schritte abgewickelt, die zum Forschungsergebnis führen. Im zweiten Kapitel wird der aktuelle Stand der Literatur zum Forschungsthema analysiert, um herauszufinden, wie LC/NC-Entwicklung momentan wahrgenommen wird, inwiefern der Ansatz im Vergleich zu HC-Entwicklung bevorzugt wird, für was für einen Typ Anwendungen er angewendet wird und von was für Entwickler. Darüber hinaus werden auch wichtige Begriffe im Bereich der Softwareentwicklungen genannt und erklärt. Im dritten Kapitel wird die Methodik bzw. das Forschungsdesign erläutert und es wird näher darauf eingegangen, wie die Idee von LexiCode einerseits mit HC und andererseits mit LC/NC umgesetzt wird und mit welchen konkreten Technologien bei beiden Ansätzen gearbeitet wird. Im darauffolgenden Kapitel, im 4. Kapitel, werden die Forschungsergebnisse vorgestellt und genau erläutert, welche der angeforderten Komponenten von LexiCode mit welchem der beiden Entwicklungsansätzen inwiefern oder überhaupt umgesetzt werden konnten. Im nächsten Abschnitt, der Diskussionsabschnitt, werden die Ergebnisse der Forschung kurz zusammengefasst, interpretiert und mit den Forschungserwartungen verglichen. Zusätzlich werden die Beschränkungen der Forschung besprochen. Im letzten Kapitel und abschließenden Kapitel wird auf die Forschung kritisch reflektiert und es werden Ideen für weiterführende Forschungen empfohlen.

2. Theoretische Fundierung

Im Folgenden werden relevante Beiträge zum Diskurs über LC/NC-Entwicklung vorgestellt. Die Beiträge unterscheiden sich einerseits nach der Umsetzung, weil manche von denen konkrete Fallstudien präsentieren, während andere rein theoretisch sind und nur die Literatur von anderen Autoren bzw. Forschern, die sich mit dem Thema befasst haben, vergleichen. Die berücksichtigte Literatur stammt aus den vergangenen drei Jahren, da die LC/NC-Entwicklung erst in dieser Zeitspanne etwas mehr an Bedeutung zugewonnen hat.

Die Auswahl der Beiträge erfolgte nach einer einfachen Suche nach dem Begriff *Low-Code Entwicklung* und ergab auch Quellen, die sich zwar auf No-Code Entwicklung konzentrieren, allerdings über KI-Chatbots und andere ähnliche Technologien. Da die Entwicklung mittels solcher Technologien keinen Einsatz in dieser Forschung finden, wurden solche Quellen auch nicht berücksichtigt.

2.1 Terminologie und Definitionen

Nachfolgend werden einige wichtige Begriffe definiert, die im Laufe der Forschung auftreten werden.

**Citizen Developer**: Anwender, die über formale Programmierkenntnisse nicht verfügen (Di Ruscio et al., 2022, S. 437).

**Fusion Development**: Softwareentwicklungsmodell, bei dem sich professionelle, ausgebildete Softwareentwickler sich auf komplexe Aufgaben wie die Entwicklung eines Backends oder die Integration in andere Systeme konzentrieren, während Citizen Developer für die Erstellung einer passenden grafischen Benutzeroberfläche zuständig sind (Elshan & Binzer, 2024, S. 1077).

**LCAP**:

**Power User**: Benutzer, die sich grundlegend mit Datenkonzepten und IT auskennen und daher geeignet sind, digitale Lösungen zu erstellen, ohne tiefgehende Programmierkenntnisse zu besitzen (Elshan & Binzer, 2024, S. 1075).

2.2 Relevante Theorien und Konzepte

Ein Konzept, welches für die vorliegende Forschung Relevanz hat, ist das magische Dreieck des Projektmanagements bzw. der sog. *Iron Triangle* (Atkinson, 1999, S. 338). Dieses Dreieck umfasst die Dimensionen Kosten, Zeit und Qualität und hat eine Bedeutung für die Forschung, da der Vergleich der beiden Entwicklungsansätze auch ein Projekt darstellt, für welches Ressourcen wie Zeit eingesetzt werden und eine gewisse Qualität des Endergebnisses erwartet wird:

Abb. 1: Iron Triangle.

Ein Bild, das Reihe, Screenshot, Dreieck, Astronomie enthält.

KI-generierte Inhalte können fehlerhaft sein.

Quelle: Eigene Darstellung auf der Basis von Atkinson, 1999.

Baumgarten et al. (2024) haben für ihre Forschung ein Modell eingesetzt, welches auf dem *Iron Triangle* von Atkinson basiert und dieses sogar erweitert. Sie bilden in ihrer Forschung ein sog. Magisches Viereck ab, welches neben den drei Dimensionen des Dreiecks noch die vierte Dimension der Flexibilität einschließt und somit ein sog. *magisches Viereck* darstellt (Baumgarten et al., 2024, S. 1213; S. 1217):

Abb. 2: Magisches Viereck



Quelle: Eigene Darstellung auf der Basis von Baumgarten et al., 2024.

Die von Baumgarten et al. (2024) durchgeführte Forschung

6. Fazit

IV. Literaturverzeichnis

Atkinson, R. (1999): *Project management: cost, time and quality, two best guesse and a phenomenon, ist time to accept other succes criteria*. Elsevier Science Ltd.

Ajimati, M. O., Carroll, N. & Maher, M. (2025. Adoption of low-code and no-code development: A systematic literature review and future research agenda. The Journal of Systems & Software, 225.

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.

Di Ruscio, D., Kolovos, D. & De Lara, J. (2022): Low-code development and model-driven engineering: two sides of the same coin?

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