

**Bachelorarbeit**

IU Internationale Hochschule für angewandte Wissenschaften

B. Sc. Wirtschaftsinformatik

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

Evsin Rahmiev

Einschreibungsnummer: 32105477

Akademiestr. 6

68159 Mannheim

Betreuer: Prof. Dr. Thorsten Fröhlich

Datum der Einreichung: 30.04.2025

## **Abstrakt**

**Zweck:** Die vorliegende Arbeit hat die Aufgabe, die Frage zu beantworten, welche Vorteile die High-Code Entwicklung im Vergleich zur Low-Code-Entwicklung anbietet.

**Wert:** Die Studie liefert der Zielgruppe Aufschluss darüber, welche Art von Anforderungen sich effektiver über High-Code Entwicklung umsetzen lassen.

**Methoden:** Für den Vergleich zwischen den beiden Entwicklungsansätzen wird ein Case-Study mit einem A/B Test durchgeführt, der nicht nach einer Konversionsrate strebt. Damit die beiden Entwicklungsansätze verglichen werden, werden eine High-Code- und eine Low-Code-Version einer Anwendung zum Deutschlernen programmiert.

**Wichtigste Ergebnisse:** Mehr Anforderungen lassen sich über die High-Code Entwicklung realisieren.

**Schlussfolgerung:** Obwohl das Programmieren mancher Anforderungen an die Anwendung bei beide Entwicklungsansätze die Gleichen Ergebnissen erzielt haben, konnten einige Anforderungen nur über die High-Code- aber nicht über die Low-Code-Anwendung umgesetzt werden.

**Schlüsselwörter:** High-code, Low-code, Softwareentwicklung

# Inhaltsübersicht

<b>Liste der Abbildungen</b>	<b>vi</b>
<b>Liste der Tabellen</b>	<b>vii</b>
<b>Liste der Formel</b>	<b>viii</b>
<b>Liste der Abkürzungen</b>	<b>ix</b>
<b>1 Einführung</b>	<b>1</b>
1.1 Ziel der Forschung	1
1.2 Forschungsfragen	2
1.3 These	3
1.4 Wert und Zielpublikum	4
1.5 Anwendungsbereich und Einschränkungen	4
1.6 Struktur des Dokuments	6
<b>2 Literaturübersicht</b>	<b>7</b>
2.1 Hintergrundinformationen	7
2.2 Bestehende Modelle und Theorien	8
2.3 Ausgewählte Fallstudien	13
<b>3 Forschungsdesign</b>	<b>18</b>
3.1 Anforderungsanalyse	19
3.2 Spezifikation der Anforderungen	20
3.3 Technologie	23
3.4 Analyse	25
3.5 Fehlerbewertung	28
3.6 Technologische Sachzwänge	29
<b>4 Ergebnisse und Diskussion</b>	<b>32</b>
4.1 Seitennavigation	32
4.2 Manipulation der Seitenelemente	36
4.3 Anpassbares Formular	39
4.4 Inzeilige Eingabefelder	42
4.5 Sitzungsunabhängige Datenspeicherung	45

4.6	Anforderungen außerhalb der Forschungsfragen	49
4.7	Ermittlung der Testkosten	51
4.8	Bestätigung der These	55
4.9	Die Ergebnisse auf dem magischen Viereck abbilden	55
4.10	Zusammenfassung	56
<b>5</b>	<b>Schlussfolgerung</b>	<b>58</b>
5.1	Kritische Reflexion	58
5.2	Methodische Reflexion	58
5.3	Empfehlungen für die künftige Forschung	60
5.4	Ausblick	60
	<b>Referenzen</b>	<b>62</b>
	<b>Appendix A. Abbildungen</b>	<b>65</b>
A-1	Leere Zoho-Seite nach Betätigen eines HTML-Links	65
A-2	Seitennavigation über den App-Menüerstller	66
A-3	Deluge Funktionen Teil 1	67
A-4	Deluge Funktionen Teil 2	68
A-5	Deluge Funktionen Teil 3	69
A-6	HC-Kontaktformular	70
A-7	HC-Kontaktformular Code	71
A-8	Feldeigenschaften von Formularfeldern	72
A-9	HTML-Editor im Form Builder	73
A-10	HTML-NC-Kontaktformular über den Zoho Creator	74
A-11	Anpassungsoptionen für den Formular-Baustein	75
A-12	Präfix-, Suffix-, Pflichtfeld- und PBD-Angabe	76
A-13	Feldtyp-, Feldgrößen- und Beschreibungstextangabe	77
A-14	HC-Lückentest	78
A-15	LC-Lückentest im Zoho-Creator	79
A-16	HTML- und CSS-Code als getrennte Dokumente	80
A-17	HTML-Snippet mit HTML- und CSS-Code	81
A-18	Bootstrap Button im normalen Zustand	82
A-19	Bootstrap Button mit Hover-Effekt	83
A-20	HC-Kontaktformular bei 550 Pixel Bildschirmbreite	84

A-21 HC-Kontaktformular bei 350 Pixel Bildschirmbreite	85
A-22 LC-Abschlusstest bei 550 Pixel Bildschirmbreite	86
<b>Appendix B. Tabellen</b>	<b>87</b>
B-1 Aufwandsermittlungswerte bei der HC-Version von Lexicode	87
B-2 Ermittlung des wahrscheinlichen Schadens bei der HC-Version	88
B-3 Aufwandsermittlungswerte bei der LC/NC-Version von Lexicode	89
B-4 Ermittlung des wahrscheinlichen Schadens bei der LC/NC-Version	90

## Liste der Abbildungen

Abbildung 1. (Magisches Dreieck)	8
Abbildung 2. (Magisches Viereck)	9
Abbildung 3. (Tagesaufwand für die Fehlerbehebung)	12
Abbildung 4. (Link zu einer Inhaltsseite)	32
Abbildung 5. (Sidebar.Menu für globale Navigation)	33
Abbildung 6. (Input-Feld für Übungsantwort gefolgt von einem Prüf-Button)	36
Abbildung 7. (Input-Feld für E-Mail-Adressen)	39
Abbildung 8. (Eigendefinierte CSS-Klasse für das Kontaktformular)	39
Abbildung 9. (Inzeilige Eingabefelder mittels HTML)	42
Abbildung 10. (Eigene Klasse zum Entfernen von Stichpunkten)	42
Abbildung 11. (Teilaufgabe zur sitzungsunabhängigen Datenspeicherung)	45
Abbildung 12. (Funktion zur browserseitigen Speicherung von Daten)	45
Abbildung 13. (Abrufen zuvor gespeicherter Werte)	46
Abbildung 14. (Bootstrap-Klasse für runde Ecken)	49
Abbildung 15. (Bootstrap-Klassen für Buttons)	49
Abbildung 16. (Responsiver Bootstrap-Container)	49
Abbildung 17. (Magisches Viereck mit den Ergebnissen von LexiCode)	56

## **Liste der Tabellen**

Tabelle 1. Funktionale Anforderungen	20
Tabelle 2. Nicht-funktionale Anforderungen	21
Tabelle 3. Technische Anforderungen	22

## Liste der Formel

Formel 1. ROI	11
Formel 2. Aufwand	12
Formel 3. Wahrscheinlicher Schaden	26



## **Liste der Abkürzungen**

DOM	Document Object Model
HC	High Code
IDE	Integrated Development Environment
LC/NC	Low-Code/No-Code
LCAP	Low-Code-Applikationsplattform
MDE	Model-Driven Engineering
ROI	Return on Investment
VSC	Visual Studio Code

# 1 Einführung

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.

Die Unterschiede der beiden Programmieransätze lassen sich bei der Entwicklung von LexiCode, eine Anwendung zum Deutschlernen, bemerkbar. Die Anwendung, die grob beschrieben eine reine Frontend Anwendung ist und unter anderem Theorie zu den grammatikalischen Grundlagen der deutschen Sprache übermitteln und diese durch interaktive Übungsaufgaben überprüfen soll. LexiCode kann in der Theorie sowohl über HC- als 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 den Anforderungen programmiert werden können und welcher der Ansätze bietet mehr Anpassungsmöglichkeiten für LexiCode an.

## 1.1 Ziel der Forschung

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 den geplanten Komponenten zu halten.

Es wird versucht, herauszufinden, welcher der beiden Entwicklungsansätze eine Umsetzung ermöglicht, bei der die Anwendung LexiCode über die folgenden Merkmale verfügt:

1. Interaktionen zwischen der Anwendung und den Nutzern
2. Visuelle und funktionelle Anpassungsfähigkeit
3. Dynamische Navigation zwischen den Anwendungsseiten
4. Eine betriebsabhängige Manipulation der UI-Komponenten
5. Inzeilige Eingabefelder
6. Sitzungsunabhängige Speicherung von Benutzerdaten im Browser

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 sowohl über HC- als über LC/NC-Entwicklung Hyperlinks angelegt werden, die die User 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 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 bzw. -komponenten 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 Anwendung neue Elemente wie Überschriften, Zeilen oder Bilder angeknüpft oder bestehende Elemente gelöscht werden<sup>1</sup>. Eine Unterfrage zu dieser Forschungsfrage ist, ob die gleiche Manipulation bei LC/NC vorhanden ist.

---

<sup>1</sup> (Flanagan, 2020, S. 415)

**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 umfassen.

**4. Lassen sich sowohl über HC als über LC/NC inzeilige 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 ihren Plätzen befinden.

Solche Felder sind eine Anforderung an LexiCode und stellen die Frage, ob sich diese über HC oder LC/NC einbauen lassen und wenn ja, ob sie visuell angepasst werden können wie z. B. in der Breite, in der Höhe, in der Randdichte oder die Hintergrundfarbe.

**5. Können Funktionen in beiden Versionen der Anwendung umgesetzt werden, die benutzerdefinierte Inhalte lokal und sessionsunabhängig, also sitzungsunabhängig 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 Datenbankbindung 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 der Forschung sowie eine klare Vorstellung für die Endergebnisse. Unabhängig von diesen Erwartungen, kann die Aussage formuliert werden, dass die HC-Entwicklung einen einfacheren Entwicklungsprozess ermöglicht, weil sie über mehr Werkzeuge zur Umsetzung der Anforderungen verfügt.

Unter einfacher ist gemeint, dass ein Entwicklungsprozess erwartet wird, bei dem sich die Anforderungen, die im späteren Teil dieser Forschung vorgestellt werden, sich wie geplant implementieren lassen. Dies würde bedeuten, dass die geplanten Systemkomponenten wie geplant aussehen und die angeforderte Programmlogik sich wie vorgestellt verhält.

In einer Konstellation, in der die Anforderungen nicht erfüllt werden können oder mehrere unterschiedliche Methoden und Werkzeuge ausprobiert werden müssen, bis die gewünschten Ergebnisse erreicht werden, kann die These nicht bestätigt werden.

## **1.4 Wert und Zielpublikum**

Die Ergebnisse dieser Forschung leisten einen Beitrag zur Debatte über die Effektivität zweier Entwicklungsansätze. Die verteidigte Ansicht, dass die LC/NC-Entwicklung die Zukunft der Softwareentwicklung ist, 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 ignoriert werden, sind die Kompatibilität von LC/NC-Anwendungen mit anderen Systemen und die Verfügbarkeit von Werkzeugen, die weitere Konfigurations- und Anpassungsmöglichkeiten bieten. Die Wiederverwendbarkeit des entwickelten Programmcodes, die Skalierbarkeit und Wartung sollten ebenfalls möglich sein. 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 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 kleinere Teams mit weniger hardware- und softwaretechnischen Ressourcen arbeiten, eine kleine Anwendung entwickeln möchten oder sich fragen, ob sie sich für HC oder LC/NC entscheiden sollen. Die Ergebnisse der Forschung sind für Lernanwendungen relevant, die auch andere Inhalte als eine Fremdsprache übermitteln sollen.

Die Forschung befasst sich nicht mit Prozessen und Diensten, die die Speicherung oder das Abrufen von Daten von Diensten, 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 sprengen würde. Insofern sind die Ergebnisse der Forschung für Frontend-Lösungen relevant.

Im Laufe der Forschung und v.a. in dem Kapitel der Ergebnisse wird zugunsten der HC-Entwicklung argumentiert, weil der Entwickler von LexiCode gegenüber diesem Entwicklungsansatz voreingenommen eingestellt ist. Diese Voreingenommenheit stammt aus der Berufserfahrung des Entwicklers, in der er sowohl mit HC- als auch mit LC/NC-Programmierung arbeiten konnte.

Der Entwickler hat die Erfahrung sammeln können, dass LC-Anwendungen Integrations- und Skalierbarkeitsschwierigkeiten aufbereiten, während HC-Anwendungen mit genügend Programmierkenntnissen diverse Erweiterungsmöglichkeiten bieten.

Für die vorliegende Forschung wurde keine empirische Studie durchgeführt und es wurden auf keiner anderen Art und Weise empirische Daten erzeugt. Stattdessen werden für die Analyse und die Argumentation als Daten die Ergebnisse der Entwicklung der Anforderungen an LexiCode verwendet.

Die Methodik der vorliegenden Forschung wurde absichtlich primär technisch-konzeptionell gestaltet ohne triangulierte Evidenz oder Metriken wie Entwicklungszeit. Der Vergleich zwischen der HC- und der LC/NC-Version der Anwendung basiert auf eine qualitative Analyse der beiden Anwendungen, ihrer Architektur und Funktionalitäten.

Es wurden bewusst keine quantitativen Metriken wie Entwicklungszeit oder Benutzerfeedback erhoben. Die Einschränkung resultiert aus dem Fokus der Arbeit auf der technischen Umsetzung der angeforderten Anwendung LexiCode und die Absicht der Forschung, herauszufinden, inwiefern sich die Anforderungen durch beide Entwicklungsansätze realisieren lassen.

Der innerhalb der Forschung vorgenommene A/B-Test wird nur von dem eingesetzten Entwickler evaluiert, anstatt auch von anderen Personen evaluiert zu werden. Der Grund dafür ist, dass die beiden Versionen der Anwendung von einer Person bewertet werden müssen, die zumindest eine grundlegende Erfahrung als Entwickler hat, unter der UI die Funktionalität einer Anwendung und ob diese effizient und/oder effektiv ist beurteilen kann. Da die unterschiedlichen Versionen einer Software in einem A/B-Test typischerweise von Nicht-Entwicklern evaluiert werden, musste für diese Forschung eine entwicklungsaffine Person ausgesucht werden.

Bei der Evaluation der Forschungsergebnisse findet eine Einordnung in die Landschaft der bestehenden LC-Literatur statt und gleichzeitig wird subjektiv aus der Perspektive des Entwicklers gewertet. Der Grund für diese zweite Evaluationsmethode ist die berufliche Erfahrung des Entwicklers sowohl als HC-Entwickler als auch als LC/NC-Entwickler.

Während er beruflich mit der LC-Plattform *Ninox* gearbeitet hat, konnte der Entwickler bereits selbst erfahren, dass obwohl sich über LC kleinere Funktionalitäten und einfachere neue Systemkomponenten schnell umsetzen lassen, wird bei zunehmender Komplexität der Anwendung schwierig, ohne den Einsatz von Drittanbietersystemen oder HC-Entwicklung bestimmte Lösungen umzusetzen. Mit der Zeit wurden weitere herkömmliche HC-Entwickler vom Arbeitgeber des Entwicklers gesucht und diese Abhängigkeit von der HC-Technologie hat dazu beigetragen, dass er eine Voreingenommenheit bzw. ein Bias für die HC-Entwicklung hat.

## **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 welchen Typ Anwendungen er angewendet wird und von welchem Typ Entwickler. Darüber hinaus werden wichtige Begriffe im Bereich der Softwareentwicklung 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 4. werden die Forschungsergebnisse vorgestellt und genau erläutert, welche der angeforderten Komponenten von Lexi-Code 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 und abschließenden Kapitel wird auf die Forschung kritisch reflektiert und es werden Ideen für weiterführende Forschungen empfohlen.

## 2 Literaturübersicht

Im Folgenden werden relevante Beiträge zum Diskurs über LC/NC-Entwicklung vorgestellt. Die Beiträge unterscheiden sich einerseits nach der Umsetzung, weil manche von ihnen konkrete Fallstudien präsentieren, während andere rein theoretisch sind und nur die Literatur von anderen Autoren, die sich mit dem Thema befassen haben, vergleichen.

Die Auswahl der Beiträge erfolgte nach einer einfachen Suche nach dem Begriff Low-Code Entwicklung und ergab Quellen, die sich zwar auf NC-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 nicht berücksichtigt.

### 2.1 Hintergrundinformationen

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

Citizen Developer: Anwender, die über formale Programmierkenntnisse nicht verfügen.

LCAP: Anwendungen, die durch visuelles Zusammenfügen vorgefertigter Software-Bausteine erstellt und individuell angepasst werden können.

Time-to-Market: die Zeit, die vergeht, bis die Entwicklung einer Produktidee oder ein Serviceangebot reif genug ist, damit eine Platzierung des Produktes bzw. der Dienstleistung am Markt erfolgen kann.

Minimum Viable Product (MVP): der minimalst mögliche Funktionsumfang eines Produkts oder einer Dienstleistung. Das Produkt bzw. die Dienstleistung werden erst dann veröffentlicht, wenn diese Entwicklungsstufe erreicht wird. Dies lässt sich daran erkennen, dass der minimale Aufwand und das erste qualitative Feedback einen gemeinsamen Punkt erreicht haben.

Earned-value-Ansatz: Softwareentwicklungsansatz, bei dem die einzelnen Anforderungen mit Value-Points gewichtet werden, anstatt gleich behandelt zu werden. Die Anforderungen mit den meisten Wertpunkten werden vorgezogen, während die mit den wenigsten Punkten zurückgestellt werden (ähnlich zu der agilen Softwareentwicklung, bei der Anforderungen nach ihrem Wert für den Kunden abgearbeitet werden)<sup>2</sup>.

---

<sup>2</sup> (Sneed & Jungmayr, 2011, S. 192)



Value-Driven Software-Engineering (wertgetriebenes Software-Engineering): Ein Softwareentwicklungsprojekt soll wirtschaftlich gerechtfertigt werden, indem die Kosten mit dem Nutzen eines Produkts abgeglichen werden<sup>3</sup>.

## 2.2 Bestehende Modelle und Theorien

Ein Konzept welches für die vorliegende Forschung Relevanz hat, ist das magische Dreieck des Projektmanagements bzw. der sog. Iron Triangle<sup>4</sup>. Dieses Dreieck umfasst die Dimensionen Kosten, Zeit, Qualität und hat eine Bedeutung für die vorliegende Forschung, da der Vergleich der beiden Entwicklungsansätze ein Projekt darstellt. Für dieses Projekt werden Ressourcen wie Zeit eingesetzt und eine gewisse Qualität des Endergebnisses wird erwartet:

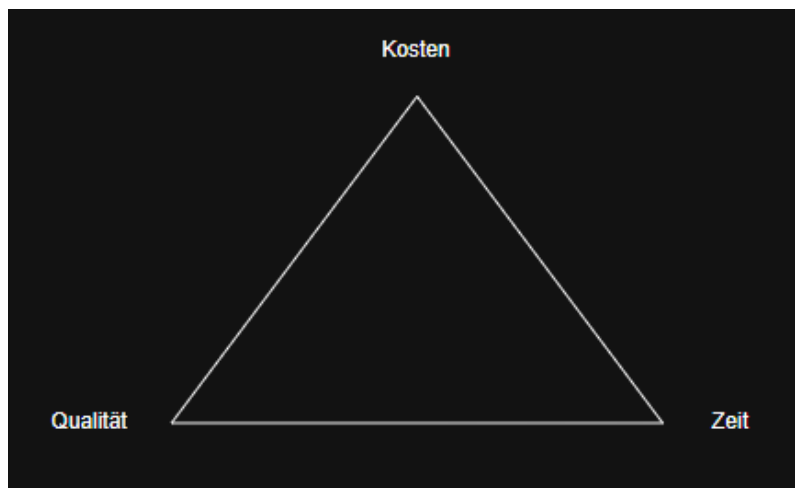


Abbildung 1. (Magisches Dreieck)<sup>5</sup>

---

<sup>3</sup> (Sneed & Jungmayr, 2011, S. 192)

<sup>4</sup> (Atkinson, 1999, S. 338)

<sup>5</sup> (Atkinson, 1999, S. 338)

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, die drei Dimensionen des Dreiecks eine um vierte Dimension der Flexibilität erweitert und somit ein sog. magisches Viereck darstellt<sup>6</sup>:

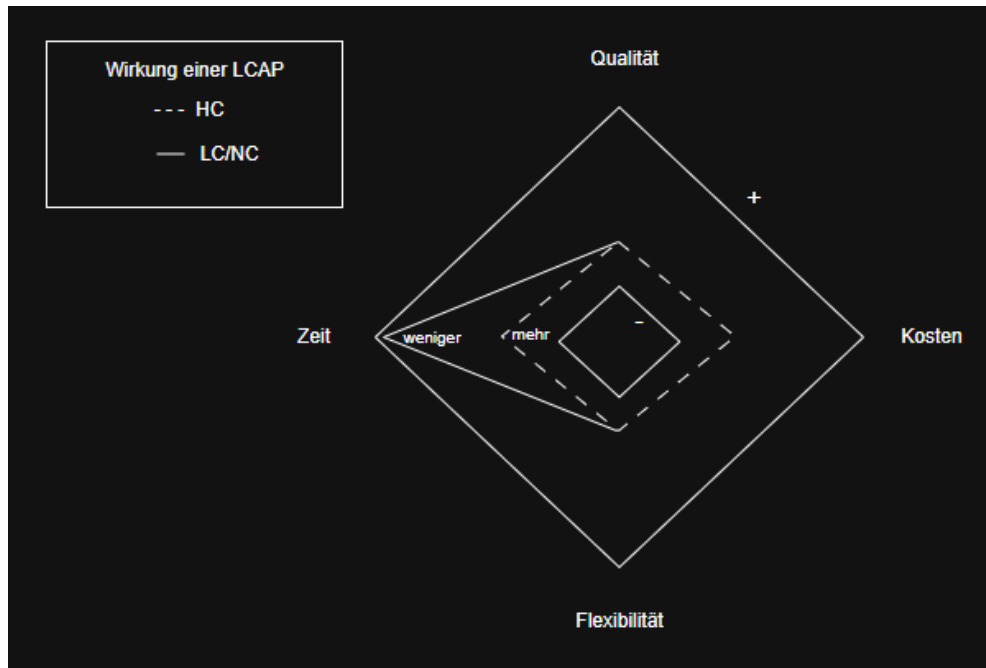


Abbildung 2. (Magisches Viereck)<sup>7</sup>

Die von Baumgarten et al. (2024) vorgestellte Abbildung des magischen Vierecks hat für die vorliegende Forschung eine abbildende Bedeutung, weil die Ergebnisse der vorliegenden Forschung auf dem magischen Viereck abgebildet werden können, um zu veranschaulichen, in welchen Dimensionen die HC-Entwicklung mehr Vorteile als die LC-Entwicklung bietet.

Auf dem magischen Viereck ist unter der Dimension der Flexibilität die Kompatibilität des jeweiligen Entwicklungsansatzes mit Drittanbietersystemen oder Schnittstellen bzw. Entwicklungsumgebungen für Backend-Lösungen wie z. B. Server oder Datenbanken zu verstehen. Solche Lösungen, Systeme und Komponenten werden im Rahmen der vorliegenden Forschung nicht entwickelt oder eingesetzt.

<sup>6</sup> (Baumgarten, Endl, & Stich, 2024, S. 1213; 1217)

<sup>7</sup> (Baumgarten, Endl, & Stich, 2024, S. 1217)

Die von Baumgarten et al. eingesetzte Abbildung des magischen Vierecks veranschaulicht die Ergebnisse der literarischen Forschung, die sie durchgeführt haben und bei der sie unterschiedliche Beiträge zum Thema LC/NC-Entwicklung berücksichtigt haben.

Diese sind z. B. Foren, in denen Low-Code bezogene Diskussionen stattfinden, eine Forschung, die nach dem Grund des Einsatzes von Low-Code fragt, eine Forschung, die die Treiber und Bremsfaktoren einer LCAP-Einführung untersucht und eine Forschung, die sich mit den technischen Herausforderungen einer LCAP-basierten Applikationsentwicklung beschäftigt<sup>8</sup>.

Je nachdem, wie die Auswirkungen der LC/NC-Entwicklung von den Autoren dieser Forschungen wahrgenommen worden sind, haben Baumgarten et al. die identifizierten Praxiserfahrungen der entsprechenden Dimension auf dem Viereck zugeordnet und mit einem Faktor von +1 auf die Dimension abgebildet, wenn die Praxiserfahrung als Verbesserung zur HC-Entwicklung wahrgenommen wurde. Mit -1 wurde eine entgegengesetzte Praxiserfahrung abgebildet und mit 0 eine neutrale<sup>9</sup>.

Auf Abb. 2. sind die Ergebnisse der von Baumgarten et al. referenzierten Forschungen geschildert, die auf Zugewinne in der Dimension der Zeit hindeuten. Dabei haben sich Baumgarten et al. darauf konzentriert, wie lang die Iterationszyklen bei der Entwicklung einer LCAP sind, wie schnell das MVP und wie schnell das Time-To-Market erreicht werden können<sup>10</sup>.

Die Ergebnisse der vorliegenden Forschung werden im späteren Kapitel über die Ergebnisse ebenfalls visuell auf dem magischen Viereck dargestellt, damit veranschaulicht werden kann, in welchen Dimensionen die LC/NC- und in welchen Dimensionen die HC-Version von LexiCode zu- oder abgewonnen hat.

Ein weiterer theoretischer Hintergrund, der die vorliegende Forschung unterstützt, beschreibt die Messung eines Softwareentwicklungsprojektes, um die (weitere) Durchführung des Projektes zu rechtfertigen. Sneed und Jungmayr heben die Vorteile des Earned-Value-Ansatzes hervor, bemängeln jedoch, dass bei diesem Ansatz die Entscheidung nicht unterstützt wird, ob ein (Teil-)Projekt überhaupt angegangen werden soll und falls ja, welche Anforderungen erfüllt werden sollen und welche nicht.

---

<sup>8</sup> (Baumgarten, Endl, & Stich, 2024, S. 1217)

<sup>9</sup> (Baumgarten, Endl, & Stich, 2024, S. 1217)

<sup>10</sup> (Baumgarten, Endl, & Stich, 2024, S. 1218)

Als Lösung bieten sie den Ansatz von Boehm<sup>11</sup>, die sog. *Value-Driven Software-Engineering*, der sich auf die wirtschaftliche Rechtfertigung der Softwareentwicklung konzentriert. Der Ansatz besagt, dass jede Aufgabe in einem Projekt nur so viel wert wie ihr wirtschaftliches Ergebnis besitzt und dieser Wert größer als die Kosten sein muss, der wiederum bedeutet, dass der ROI positiv sein muss<sup>12</sup>.

Boehm berechnet den ROI wie folgt:

$$ROI = (Nutzen - Kosten)/Kosten \quad (1)$$

Formel 1. ROI

Bei der Entwicklung einer Software ist es angesichts der Kosten wichtig, möglichst fehlerfrei zu entwickeln, da die nachträgliche Fehlerbehebung zusätzliche Kosten verursacht, die 17-26 % der gesamten Lebenszykluskosten einer Software ausmachen können<sup>13</sup>. Umfangreiches Testen der Software vor der Veröffentlichung kann zwar die späteren Kosten für die Fehlerbehebung reduzieren, verlängert allerdings die Entwicklungszeit vor der Veröffentlichung, die ebenfalls zu einem Werteverlust führt.

Daher hat jede Projektleitung abzuwägen, ob der durch die zusätzlich benötigte Entwicklungszeit verursachte oder der durch die Fehlerhaftigkeit bedingte Werteverlust größer ist<sup>14</sup>.

Es werden mehrere Ansätze empfohlen, um die Fehler bei der Entwicklung einer Software zu reduzieren und zwei von denen, die in der vorliegenden Studie angewendet wurden, sind einerseits die Wiederverwendung bestehender Softwarekomponenten, die im Fall von LexiCode Layouts, UI-Elemente wie Eingabefelder, Formulare oder Tests zur Wissensabfrage sind.

Ein weiterer Ansatz ist das wertbasierte Testen dieser Systemkomponenten, das sog. Value-Driven Testing, das als Maßnahme stark empfohlen wird, weil umfangreiches Testen in der Entwicklungsphase die Kosten für die Fehlerbehebung in der Wartungsphase reduzieren kann. Dies ist vor allem daran zu erkennen, dass die durchschnittlichen Fehlerbehebungskosten als zeitlicher Aufwand in der Testphase innerhalb der Entwicklung bis zu maximal 15 Tagen dauern kann.

---

<sup>11</sup> (Boehm, 1981, S. 597)

<sup>12</sup> (Baumgarten, Endl, & Stich, 2024, S. 193)

<sup>13</sup> (Sneed & Jungmayr, 2011, S. 193)

<sup>14</sup> (Sneed & Jungmayr, 2011, S. 195)

In der produktiven Phase eines Softwareprodukts und je nach der Größe des Betriebes können Tests mindestens 13 und im schlimmsten Falle 92 Tage dauern:

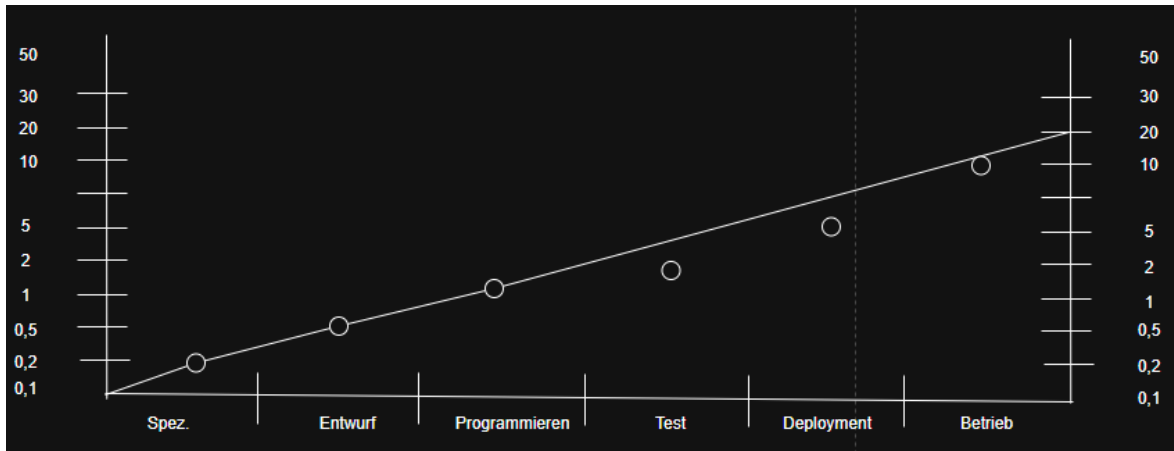


Abbildung 3. (Tagesaufwand für die Fehlerbehebung)<sup>15</sup>

Die These von Sneed und Jungmayr, dass allein das Finden von Fehlern den Aufwand für das Testen in Bezug auf den eingesparten Folgekosten rechtfertigen kann, weist eine Schwachstelle auf, weil die Wahrscheinlichkeit besteht, dass die Fehlerkosten niedriger als die Testkosten sind und somit der Testaufwand nicht gerechtfertigt ist. Daher entsteht der Bedarf, zuerst über die Information zu verfügen, wie der Testnutzen ermittelt wird und wie die Testkosten im Voraus zu berechnen sind. Um dies zu erreichen bzw. die Testkosten zu ermitteln, haben Sneed und Jungmayr die folgende Formel zusammengestellt:

$$\text{Aufwand} = AF \times \left[ \frac{TF \times TU}{TP} + (TP \times (1 - TA) TE \times \left( \frac{MT}{TB} \right)) \right] \quad (2)$$

Formel 2. Aufwand

Die wichtigsten Parameter dabei sind die Anzahl der Solltestfälle (TF), die angestrebte Testüberdeckung (TU), die manuelle Testproduktivität (TP), der Grad der Testautomatisierung (TA), die Testbarkeitsmetrik (TB), die mittlere Testbarkeit (MT), der Testskalierungsexponent (TE) und der Justierungsfaktor (AF)<sup>16</sup>.

<sup>15</sup> (Sneed & Jungmayr, 2011, S. 198)

<sup>16</sup> (Sneed & Jungmayr, 2011, S. 206)

Das Ergebnis ermittelt eine geschätzte Anzahl von 700 Personentagen, die nötig wäre, um das System während der Entwicklung und vor der Veröffentlichung zu testen<sup>17</sup>.

Darüber hinaus wird es für sinnvoll gehalten, den Test-ROI zu ermitteln. Für den Test-ROI ist die Anzahl der erwarteten Fehler zu berücksichtigen, für die ein Wert von 6 Fehlern pro 1000 Zeilen Code, also insgesamt 600 Fehler angenommen wird<sup>18</sup>. Laut einer weiteren Annahme werden mindestens zwei Drittel der vorhandenen Fehler durch systematische Tests aufgedeckt, die die Fehleranzahl auf 200 reduzieren und der größere Anteil der 400 Fehler nicht erst in der Wartungsphase entfernt werden muss.

Die Fehler lassen sich nach Ursprung wie folgt verteilen: 40% Anforderungsfehler, 30 % Entwurfsfehler und 30 % Codefehler. Bei der Behebung dieser Fehler wird mit einem geschätzten Aufwand nach Personentagen gearbeitet, der folgendermaßen abläuft: 2,5 Personentage für Anforderungsfehler, 1,0 Personentag für Entwurfsfehler und 0,5 Personentage für Codefehler<sup>19</sup>.

Die Berechnung des geschätzten Aufwands für die jeweilige Fehlerquelle ergibt die Fehlerbehebungskosten, die zusammen mit den Kosten des wahrscheinlichen Schadens als Summe den Einsparungspotenzial ergeben, aus dem die Kosten des Testaufwands (Personentage x Tagessatz) subtrahiert und noch einmal durch die Testkosten dividiert werden<sup>20</sup>:

$$\text{ROI} = (\text{Nutzen} - \text{Kosten}) / \text{Kosten}.$$

## 2.3 Ausgewählte Fallstudien

Für die vorliegende Forschung spielte es eine Rolle, bestehende Literatur zum Thema HC- und LC/NC-Entwicklung zu berücksichtigen. Zu der berücksichtigten Literatur zählen weitere eigene experimentenbasierte Forschungen und Literaturanalysen.

---

<sup>17</sup> (Sneed & Jungmayr, 2011, S. 207)

<sup>18</sup> (Sneed & Jungmayr, 2011, S. 206)

<sup>19</sup> (Sneed & Jungmayr, 2011, S. 207)

<sup>20</sup> (Sneed & Jungmayr, 2011, S. 207)

Ajimati, Carroll und Maher (2025) beschäftigen sich mit der Einführung von LC/NC-Technologien, die Bedeutung von Citizen Developern und deren Beitrag zur digitalen Transformation. Sie analysieren die Vorteile der LC/NC-Entwicklung wie die Effizienzsteigerung und die Herausforderungen wie Sicherheitsrisiken. Sie entwickeln ein theoretisches Rahmenwerk und identifizieren Forschungslücken für zukünftige Untersuchungen<sup>21</sup>.

Aveiro, Freitas, Cunha, Quintal und Almeida (2023) vergleichen die HC- und LC/NC-Entwicklung anhand des Systems nexusBRaNT, das in einem Forschungsprojekt für kognitive Rehabilitation eingesetzt werden soll<sup>22</sup>. In der Studie wird festgestellt, dass durch LC der Arbeitsaufwand stark reduziert und die Systemkomplexität verringert wird. Dies ermöglicht Unternehmen, effizienter und flexibler Anwendungen zu entwickeln.

Böhler (2023) thematisiert die Revolution in der Softwareentwicklung durch den LC/NC-Ansatz. Er argumentiert, dass die LC-Technologie die Entwicklungszeit verkürzt, IT-Ressourcen entlastet und Unternehmen mehr Flexibilität bietet. Er setzt sich damit auseinander, dass Nicht-Programmierer durch LC-Entwicklung Anwendungen erstellen und somit in Unternehmen die Effizienz gesteigert und die Digitalisierung beschleunigt wird<sup>23</sup>.

Di Ruscio et al. (2022) vergleichen die LC-Entwicklung und das MDE und zeigen, dass beide Entwicklungsansätze Modellbasierung und Automatisierung nutzen, sich jedoch anhand der Flexibilität und der Zielgruppe unterscheiden<sup>24</sup>. Während die LC/NC-Entwicklung mehr Wert auf Benutzerfreundlichkeit und schnelle Entwicklung setzt, bietet MDE mehr Anpassungsfähigkeiten und erfordert tiefergehendes Fachwissen.

Elshan und Binzer (2024) zeigen in ihrer eigenen Studie, dass LC/NC-Plattformen die digitale Transformation unterstützen, indem sie für eine sog. Demokratie in der Softwareentwicklung sorgen, Innovationsprozesse beschleunigen und die Arbeit der IT-Abteilungen erleichtern. Obwohl bei der LC-Technologie gewisse technische Einschränkungen und Governance-Herausforderungen vorhanden sind, können Unternehmen durch transparente Strategien, Schulungen und KI-Integration, die in der vorliegenden Forschung nicht näher behandelt wird, das Potenzial dieser Technologie vollständig ausschöpfen<sup>25</sup>.

---

<sup>21</sup> (Ajimati, Carroll, & Maher, 2025, S. 2)

<sup>22</sup> (Aveiro, Freitas, Cunha, Quintal, & Almeida, 2023, S. 2)

<sup>23</sup> (Böhler, 2023)

<sup>24</sup> (Di Ruscio, et al., 2022, S. 440)

<sup>25</sup> (Elshan & Binzer, 2024, S. 1078)

Bärmann (2023) erklärt, dass mittels LC/NC-Entwicklung ERP-Systeme schneller und flexibler angepasst und implementiert werden können, um maßgeschneiderte Lösungen effizienter zu realisieren<sup>26</sup>. Damit LC/NC-Plattformen erfolgreich eingesetzt werden, müssen klare Ziele definiert und die Plattformen sorgfältig gesteuert werden. Dadurch können unkontrollierte Anpassungen und Ineffizienzen vermieden werden.

Gialitakis, Tsakalidis, Nousias und Vergidis (2024) fokussieren sich auf die Beschleunigung der Prototypenerstellung prozessorientierter Anwendungen über LC-Plattformen im griechischen öffentlichen Sektor und die Effizienzsteigerung dabei. Sie sind der Überzeugung, dass LC als Lösung viel Potenzial für öffentliche Einrichtungen verspricht und digitale Transformationsziele flexibler und kosteneffizienter erreicht<sup>27</sup>.

Hensen (2023) erläutert die schnellere und kosteneffiziente Softwareentwicklung durch LC/NC-Plattformen, fördert die Zusammenarbeit zwischen der IT und den Fachabteilungen und betont, dass trotz der Vorteile der LC-Entwicklung klare Governance und Integrationsstrategien erforderlich sind. Laut seiner Forschung hängt eine erfolgreiche Implementierung von Schulungen, strategischer Planung und kontinuierlicher Evaluierung ab. Nur so können Qualität und Sicherheit gewährleistet werden<sup>28</sup>.

Theam Lim et al. (2024) haben die Zugänglichkeit von LC/NC-Plattformen für Lerner untersucht und welchen Lernpotenzial sie bieten. Den Ergebnissen ist zu entnehmen, dass aus diesen Plattformen die Aneignung von Softwareentwicklungskompetenzen erleichtert werden kann<sup>29</sup>.

Liu, Jiang, Guo, Chen und Qiao (2023) analysieren 974 aufgefundene Fehler in den vier LC/NC-Plattformen OutSystems, Mendix, Appsmith und Budibase und identifizieren häufige Ursachen und Auswirken der Bugs. Laut den Ergebnissen treten 60 % der Fehler in der Design- und Spezifikationsphase auf, mehr als 37 % der Bugs verursachen unerwartetes Verhalten ohne klare Fehlermeldungen und UI-Fehler sind besonders problematisch, da LC-Plattformen spezifische Merkmale haben<sup>30</sup>.

---

<sup>26</sup> (Bärmann, 2023, S. 50)

<sup>27</sup> (Gialitakis, Tsakalidis, Nousias, & Vergidis, 2024, S. 655)

<sup>28</sup> (Hensen, 2023, S. 36)

<sup>29</sup> (Theam Lim, et al., 2024, S. 490)

<sup>30</sup> (Liu, Jiang, Guo, Chen, & Qiao, 2024, S. 695)



Schreiner (2024) stellt in seiner Forschung die verkürzten Entwicklungszeiten und den reduzierten Bedarf an spezialisierten Entwickler durch die visuellen Entwicklungswerkzeugen und vorgefertigte Komponenten der LC-Plattformen. Er ist der Ansicht, dass Unternehmen von einer höheren Flexibilität, schnelleren Anpassungen an Geschäftsanforderungen und einer beschleunigten Markteinführung ihrer Anwendungen profitieren<sup>31</sup>.

Krogh und Chun (2025) befassen sich mit dem unternehmerischen Einsatz von Citizen Developern, LC-Plattformen und generativer KI, um die digitale Transformation zu beschleunigen und IT-Abteilungen zu entlasten. Sie sind überzeugt, dass dabei echter Business Value entsteht, weil fachliche und technische Teams gemeinsam in einem klar definierten Rahmen Innovationen entwickeln<sup>32</sup>.

Phalake, Joshi, Rade, Phalke und Molawade (2024) untersuchen, wie Optimierungstechniken in LC-Plattformen zu nachhaltigeren und ressourcenschonenderen Softwarelösungen führen können, v. a. durch die Reduktion der Ausführungszeit und des Speicherbedarfs<sup>33</sup>. In der Fertigungsindustrie ist das Potenzial sonderlich hoch, komplexe datenbasierte Prozesse effizienter zu gestalten und gleichzeitig den Entwicklungsaufwand zu minimieren.

Novales und Mancha (2023) argumentieren dafür, dass Unternehmen mithilfe von Citizen Developern und LC-Plattformen ihre digitale Transformation schneller vorantreiben können, indem sie Fachabteilungen in die Lage setzen, eigenständig Softwarelösungen zu entwickeln. Dabei sind Governance, Schulung und strategische Planung von Bedeutung, um Innovation mit Kontrolle und Sicherheit erfolgreich kombinieren zu können<sup>34</sup>.

Gomes und Brito (2022) beschreiben die Vorteile von LC-Plattformen wie schnellere Entwicklung, Benutzerfreundlichkeit, Kosteneffizienz und ihre Nachteile wie die eingeschränkte Anpassbarkeit oder Leistungsschwierigkeiten<sup>35</sup>. In der Studie wird erklärt, dass LC-Plattformen ein bedeutsames Werkzeug zur Demokratisierung der Softwareentwicklung und zur Förderung der digitalen Transformation in verschiedenen Anwendungsbereichen sind.

---

<sup>31</sup> (Schreiner & F., 2024, S. 44)

<sup>32</sup> (Krogh & Chun, 2025, S. 217)

<sup>33</sup> (Phalake, Joshi, Rade, Phalke, & Molawade, 2024, S. 5717)

<sup>34</sup> (Novales & Mancha, 2023, S. 222)

<sup>35</sup> (Gomes & Brito, 2022, S. 2)

Pichidtienthum, Pugsee und Cooharajanane (2021) stellen ein LC-basiertes Modul für das ERP-System *Odoo*, das die Entwicklungszeit durch die automatisierte Erzeugung von Programmcode um durchschnittlich 20 % reduziert. Die Kombination von Low-Code-Konzepten und Automatisierungssystemen steigert die Effizienz bei der Erstellung funktionaler Odoo-Module, insbesondere für Citizen Developer<sup>36</sup>.

Sahay, Indamutsa, Di Ruscio und Pierantonio (2020) vergleichen acht LC-Plattformen in Bezug auf ihre Funktionen und Dienste. Durch die Analyse dieser Plattformen ermöglichen die Autoren eine fundierte Auswahlentscheidung für Entwickler und Unternehmen entsprechend ihren individuellen Anforderungen<sup>37</sup>.

Brandt-Pook (2020) beschreibt den Prozess der Softwareentwicklung und leistet einen praxisnahen Überblick über den gesamten Prozess der Softwareentwicklung – von der ersten Idee bis zum Betrieb des fertigen Systems. Das Buch stellt nicht das Programmieren selbst, sondern die vorbereitenden Tätigkeiten und die Zusammenarbeit zwischen IT-Fachleuten und Auftraggebern in den Mittelpunkt, wodurch eine wertvolle Grundlage für die Durchführung eigener Projekte entsteht<sup>38</sup>.

Die in diesem Unterkapitel kurz formulierten Zusammenfassungen der Literatur wurden über den KI-Chatbot ChatGPT generiert, damit nur die wichtigsten Aspekte der ausgewählten Fallstudien grob erwähnt werden können. Anstatt ungeändert in der vorliegenden Forschung übernommen zu werden, wurden die Zusammenfassungen zuerst umformuliert. Im späteren Kapitel 4. über die Ergebnisse und die Diskussion wurden die Literaturquellen sowohl über ChatGPT als auch manuell zusammengefasst, damit die Inhalte, die von der KI ausgelassen wurden, auch berücksichtigt werden können.

---

<sup>36</sup> (Pichidtienthum, Pugsee, & Cooharajanane, 2021, S. 529)

<sup>37</sup> (Sahay, Indamutsa, Di Ruscio, & Pierantonio, 2020, S. 171)

<sup>38</sup> (Brandt-Pook, 2020, S. 8)

### 3 Forschungsdesign

Um die gewünschten Ergebnisse hinter der vorliegenden Forschung zu erzielen bzw. die Forschungsfragen zu beantworten und die Forschungsthese zu bestätigen, wird ein Case Study durchgeführt, das auf dem in der Softwareentwicklung bekannten A/B-Test basiert. LexiCode wurde von einer Person im Umfang der Anforderungen, die später erläutert werden, einmal als eine klassische HC-Lösung und parallel als LC/NC-Lösung umgesetzt.

Es wurde nicht darauf abgezielt, eine vollständige, ausgereifte und veröffentlichte Anwendung zu entwickeln, sondern vielmehr wurde es angestrebt, die festgelegten Anforderungen der Anwendung gemäß der beiden Entwicklungsansätze so umzusetzen, dass sie die Anforderungen erfüllen. Somit hat es gereicht, dass jede der Anforderungen, wie z. B. die lokale Navigation über Hyperlinks, in den beiden Instanzen der Anwendung nur auf einer Maske oder Seite umgesetzt wird, damit mit überschaubarem Aufwand festgestellt werden kann, ob sich die Anforderungen in der jeweiligen Entwicklungstechnologie überhaupt umsetzen lassen.

Sowohl die HC- als die LC/NC-Version von LexiCode wurden von der gleichen Person umgesetzt, da die Anforderungen an LexiCode innerhalb dieser Forschung von der Anzahl und der Schwierigkeit der Umsetzung her überschaubar sind und somit sich die Beteiligung von weiteren Entwicklern erübrigt hat.

Die Implementierung durch einen einzigen Entwickler hat den Kommunikationsaufwand auf null reduziert und dadurch mehr Zeit zur Umsetzung des Projekts gewonnen, weil ein großer Teil der Arbeit in einem Team aus Kommunikation besteht, die nicht immer produktiv ist. Die Zeit, die in den Entwicklungsprozess eingeflossen ist, wurde für diese Forschung nicht gemessen, da es nicht vorhersehbar war, ob sich alle Anforderungen überhaupt umsetzen lassen.

Die in den nächsten Kapiteln beschriebenen Anforderungen wurden so konzipiert, dass sie eine grundlegende Digitalisierung des Prozesses des Deutschlernens ermöglichen, aber sich an den Erfahrungs- und Kenntnisstand des Entwicklers richten, damit die Entwicklungszeit sich nicht durch den Erwerb neuer Programmierkompetenzen zu stark verlängert.

Das Konzept des A/B-Testens wurde an die Bedingungen der vorliegenden Forschung angepasst. Anstatt die beiden Versionen von LexiCode Probanden vorzustellen, um dann anhand der Konversionsrate dieser Probanden die Effektivität beider Systeme zu vergleichen<sup>39</sup>, werden innerhalb der vorliegenden Forschung die beiden Anwendungsversionen von dem Entwickler verglichen. Dadurch soll die Frage beantwortet werden, ob sich eine leistungsfähigere Anwendung mittels HC oder LC/NC bauen lässt.

Da der A/B-Test zu einem zunehmend allgegenwärtigen und kritischen Teil von Onlinegeschäften geworden ist<sup>40</sup> und LexiCode mit etwaigem Ausbau das Potenzial hat, kommerziell bei einer hohen Nutzerzahl eingesetzt zu werden, eignet sich der A/B-Test als Forschungsmethodik.

Des Weiteren eignet sich das A/B-Testen, insbesondere das Werk von Siroker für die Methodik der vorliegenden Forschung, weil sie in ihrer Herausgabe die Bedeutung grundlegender Komponenten vieler Anwendungen und Websites, wie z. B. Calls-to-Action<sup>41</sup>, Inhalte, visuelle Medien, Seitennavigationen und Formulare, erklären und zu jeder Komponente Ratschläge liefern, wie diese am besten gebaut und getestet werden kann.

### 3.1 Anforderungsanalyse

Die Anforderungen, die LexiCode als Anwendung zum Lernen von Deutsch zu erfüllen hat, richten sich nach dem typischen Lernweg von Menschen, die nicht nur die deutsche Sprache, sondern jede für sie fremde Sprache beherrschen wollen. Diese potenziellen Lernenden stellen nach dem Entwickler die wichtigsten Stakeholder von LexiCode dar. Dieser soz. *typischer* Lernweg ist bei allen Lernenden hinsichtlich der Reihenfolge der gelernten Inhalte gleich.

Die gelernten Inhalte teilen sich bei dem Fremdsprachenerwerb nach Niveaus auf, variieren von Anfänger (A1) bis zu Fortgeschritten (C1) und schließen Grundlegende Sprachkompetenzen ein, wie einfache Begrüßungen bis zum Schreiben komplexer Aufsätze.

Demnach muss LexiCode so konstruiert sein, dass die Anwendung ihren Nutzern einen Lernbereich bereitstellt, in dem die Lerninhalte nach Niveau aufgeteilt sind und über Praxisaufgaben geübt werden können, die idealerweise eine Bewertung auf die Leistung liefern.

---

<sup>39</sup> (Siroker, 2013, S. 8)

<sup>40</sup> (Siroker, 2013, S. 8)

<sup>41</sup> (Siroker, 2013, S. 22)

Da in konventionellen Lehrbüchern durch die einzelnen Seiten einfach durchgeblättert werden kann, muss LexiCode ermöglichen, dass Benutzer mit wenig Anstrengung die Inhalte durchstöbern können. Die Schreibfelder in jedem Lehrbuch befähigen die Lernenden ihre Notizen, Vermerke und Übungsantworten direkt im Lehrbuch zu hinterlassen – eine Eigenschaft, die in LexiCode vorhanden sein sollte.

Fragebögen und Übungen, bei denen die Antworten in Schreibfeldern innerhalb der Fragezeile eingetragen werden, stellen häufige Aufgabenarten in herkömmlichen Lehrbüchern, die entsprechend von einer Anwendung zum Deutschlernen bereitgestellt werden sollen.

Die bereits erwähnten Vermerke, die in einem Lehrbuch hinterlassen werden, werden von den Lernenden wiederholt aufgeschlagen, um Gelerntes zu wiederholen oder Übungen neu zu bearbeiten. Daher müssen diese schriftlichen Informationen in LexiCode zu jeder Zeit abrufbar sein.

## 3.2 Spezifikation der Anforderungen

Die in dem letzten Kapitel aufgezählten Anforderungen lassen sich konkretisieren und in den Unterkategorien der funktionalen, der nicht-funktionalen und der technischen Anforderungen untergliedern. Die funktionalen Anforderungen beschreiben die spezifischen Funktionen und Verhaltensweisen, die die Software erfüllen muss:

Tabelle 1. Funktionale Anforderungen

Nr.	Anforderung	Beschreibung
F1	Lokale und globale Navigation	Benutzer können zwischen den Haupt- und Unterseiten der Anwendung navigieren.
F2	Interaktivität	Benutzer können aktiv mit der Anwendung interagieren.
F3	Dateneingabe und -auswertung	Benutzer können Daten eingeben, die auch ausgewertet werden.
F4	Datenspeicherung	Auf der Anwendung können vom Benutzer eingegebene Daten gespeichert werden.
F5	Ergonomische visuelle Anpassbarkeit	Die Anwendung muss für die Benutzer ergonomisch visuell anpassbar sein.

F1 muss sicherstellen, dass die Benutzer von LexiCode über Navigationsmenüs von einer Inhaltsseite zu einer anderen navigieren können. Diese Navigation soll uneingeschränkt funktionieren und optimalerweise über Hyperlinks erfolgen.

Unter der Anforderung F2 ist zu verstehen, dass die Anwendung den Nutzern nicht nur ermöglicht, Inhalte statisch lesen zu können, sondern diese zusätzlich zu ergänzen und zu erweitern. Diese Anforderung lässt sich mit der nächsten Anforderung, F3, weiter spezifizieren, da F3 leisten soll, dass die Benutzer auf der Anwendung Daten eingeben können. Diese Daten sind Antworten zu den Übungsaufgaben, die von LexiCode bewertet werden sollen.

Die Datenspeicherung, die mit F4 angefordert wird, soll nur die Speicherung der Übungsantworten ermöglichen. Diese Speicherung soll unabhängig von einem Server erfolgen, weil dies nicht im Rahmen der vorliegenden Case Study liegt. Die gespeicherten Daten sollen bei späteren Lernvorgängen weiterhin aufrufbar sein und in unveränderter Form vorliegen.

F5 soll die User-Experience bequemer gestalten, indem die Benutzer die Möglichkeit haben, den Hintergrund der Anwendung auf einer dunklen Farbe umzuschalten und somit den sog. *Dark Mode* zu aktivieren. Diese Umschaltung wird per Button-Druck erfolgen.

Die nachfolgend beschriebenen nicht-funktionalen Anforderungen beschreiben die Qualität von LexiCode in Bezug auf Benutzerfreundlichkeit und Zweckmäßigkeit:

Tabelle 2. Nicht-funktionale Anforderungen

Nr.	Anforderung	Beschreibung
NF1	Responsivität	Die Anwendung muss sich an die Bildschirmauflösung unterschiedlicher Endgeräte anpassen können.
NF2	Intuitive Benutzeroberfläche	Die Benutzeroberfläche muss so gestaltet sein, sodass sie unabhängig von der technischen Erfahrung der Nutzer einfach zu bedienen ist.
NF3	Optisch ansprechbare Oberfläche	Die Benutzeroberfläche muss mit modernen Entwicklungstechnologien visuell gestaltet sein.
NF4	Lehrbuchnähe	Die Anwendung muss von der von dem Design und der Funktionalität her einem konventionellen Lehrbuch ähneln.
NF5	Übersetzung auf Englisch	Die auf der Anwendung vorhandenen Lerninhalte müssen auch auf Englisch übersetzt sein.

Die Responsivität in Anforderung NF1 soll zum Ergebnis führen, dass die Anwendung, die als eine Desktop-First Anwendung, d. h. primär für die Nutzung mit Desktop-Rechnern und Laptops, entwickelt wird. LexiCode soll sich ebenfalls an den Bildschirmauflösungen kleinerer mobilen Endgeräte wie Smartphones oder Tablets so anpassen, dass die Inhalte noch gut ersichtlich und lesbar sind.

Da die gezielte Nutzergruppe aus Menschen bestehen soll, die bereits Fremdsprachen gelernt haben oder erst mit LexiCode eine lernen werden, soll die Anwendung nach NF2 intuitiv zu bedienen sein. Somit soll sich die Anwendung von großen Systemen unterscheiden, die oftmals intensive Einarbeitungen voraussetzen, bevor sie produktiv verwendet werden können.

NF3 fordert an, dass die Anwendung auf der Benutzeroberfläche eine gute Kombination aus einfachem, nachvollziehbarem Design, geeigneten Farben, passender Schriftgröße und überlegter visueller Hierarchie bietet.

In NF4 wird angefordert, dass LexiCode vom Design und von dem Benutzererlebnis her an den herkömmlichen, gedruckten Lehrbüchern erinnert, da dieses Medium meistens von Sprachlernenden bereits eingesetzt worden ist. Somit wird verlangt, dass die Theorieinhalte und Praxisübungen so aussehen und strukturiert sind, wie sie in Lehrbüchern in Papierform zu finden sind.

Die letzte, nicht-funktionale Anforderung, NF5, hat den Zweck, Lernenden unabhängig von ihrem Hintergrund zu ermöglichen, mit LexiCode arbeiten zu können. Da Englisch die am meisten gelernte Fremdsprache ist, werden Benutzer mit LexiCode lernen können, da die Inhalte auf Englisch übersetzt sein werden.

Die Gruppe der technischen Anforderungen bezieht sich auf die Technologien, Werkzeuge und Architekturen, die für die Entwicklung der Software verwendet werden:

Tabelle 3. Technische Anforderungen

Nr.	Anforderung	Beschreibung
T1	Webtechnologie	Für die technische Umsetzung sind gängige Webentwicklungstechnologien wie HTML, CSS und JavaScript einzusetzen.
T2	Frameworks	Zur Sicherstellung einer schnellen und vereinfachten Entwicklung sollen gängige Frameworks wie Bootstrap angewendet werden.
T3	IDE	Die Entwicklung soll innerhalb einer integrierten Entwicklungsumgebung stattfinden.
T4	Aktualität der Werkzeuge	Die für die Entwicklung eingesetzten Werkzeuge sollen auf dem neuesten Stand und die erforderlichen Updates durchgegangen sein.
T5	Customization	Die umzusetzenden UI-Elemente müssen entwicklerseitig flexibel und visuell gestaltbar sein.

Anforderung T1 wurde festgelegt, weil LexiCode als Webanwendung über einen Browser zugänglich sein soll, ohne dass Nutzer die Anwendung auf ihre Rechner installieren müssen. Die Entwicklung von LexiCode als Webanwendung wird für zukünftige Projekte ermöglichen, dass sie als eine native Applikation für mobile Endgeräte erweitert werden kann.

Durch T2 wird verlangt, dass bei der Entwicklung Frameworks eingesetzt werden, weil diese schnelleren Lösungen bei der Entwicklung ermöglichen. Solche Lösungen sind z. B. die Einstellung der Textausrichtung, Innen- und Außenabstände der UI-Elemente und UI-Elemente responsiv machen.

Die Entwicklung soll laut T3 in einer integrierten Entwicklungsumgebung (*IDE*) stattfinden, da diese durch diverse Eigenschaften den Entwicklungsprozess erleichtern. Diese Eigenschaften sind z. B. farbliche Hervorhebung der Code-Schlüsselwörter, Ergänzungsvorschläge und verkürzte Schreibweise von Code-Befehlen.

Das Schreiben von Programmcode ist über reine Texteditoren wie Notepad oder MS Office Word möglich, aber da diese über die aufgezählten Eigenschaften einer *IDE* nicht verfügen, werden sie in diesem Case Study für die Entwicklung nicht eingesetzt.

Die eingesetzten Werkzeuge sollen nach T4 auf einem aktuellen Stand sein, weil die aktuellen Versionen von Programmiersprachen, Frameworks und *IDEs* alle Standardfeatures weiterhin unterstützen und sogar neue anbieten, die die Entwicklung weiter vereinfachen.

T5 sorgt dafür, dass bei der Entwicklung ausreichend Möglichkeiten zur visuellen und funktionalen Anpassung vorhanden sind.

### 3.3 Technologie

Die im Rahmen der vorliegenden Forschung eingesetzten Technologien lassen sich in zwei Bereiche unterteilen: die Technologien, die für die Entwicklung der HC-Version von LexiCode herangezogen wurden, und die Technologien, die für die LC/NC-Version verwendet wurden. Um die HC-Version zu verwirklichen, wurden die gängigen Webentwicklungstechnologien HTML, CSS und JavaScript ausgewählt.

Die Verwendung von HTML begründet sich dadurch, dass die Markup-Sprache<sup>42</sup> Elemente für viele gängige Anwendungskomponenten bietet wie Formulare, Listen, Texte, Buttons, Überschriften und andere. Zudem können über HTML semantische Elemente wie `<nav>` für Navigation, `<article>` für Artikel oder `<section>` in Anspruch genommen werden<sup>43</sup>, die die Nachvollziehbarkeit des Codes erleichtern und den gebauten Elementen eine semantische Bedeutung zuweisen.

---

<sup>42</sup> (Bühler, Schlaich, & Sinner, 2017, S. 18)

<sup>43</sup> (Bühler, Schlaich, & Sinner, 2017, S. 15)



Da mittels HTML gestaltete Anwendungen mit CSS ergänzt werden, wurde logischerweise CSS in dem vorliegenden Projekt verwendet. CSS bietet vielfältige Anpassungsmöglichkeiten für HTML-Elemente inklusive Breite und Höhe der Elemente, Zeilenabstände der Texte, Schriftgröße, Farben, Schatteneffekte, Animationen und viele andere. Dieses breite Spektrum an Anpassungsmöglichkeiten eignen CSS zur Gestaltung einer ansprechbaren und modernen Benutzeroberfläche.

Damit LexiCode Funktionen ausführen kann, wurde die Funktionslogik der Anwendung über die Programmiersprache JavaScript geschrieben, die oft in Kombination mit HTML und CSS verwendet wird und aufgrund ihrer modernen ES6 Eigenschaften einen vereinfachten Codesyntax wie Pfeilfunktionen ermöglicht<sup>44</sup>.

Des Weiteren lässt sich über JavaScript die bereits angekündigte DOM-Manipulation über *Event Listener*<sup>45</sup> realisieren, die aufgrund ihrer Möglichkeiten, die UI-Elemente zu klonen, löschen oder neu zu erstellen, ein mächtiges Werkzeug der Webentwicklung ist, und aus diesem Grund JavaScript häufig den Ruf eines bevorzugten Webentwicklungstools verleiht.

Die Entwicklung der HC-Version von LexiCode erfolgt in der Entwicklungsumgebung Visual Studio Code, die viele gängige Projektverwaltungsaufgaben Funktionen zur Verfügung stellt. Diese sind z. B. das Erstellen von neuen Projektordnern und Dokumenten, die Verknüpfung Versionsverwaltungsplattformen wie *Git*, die Ausführung von Terminal<sup>46</sup> Befehlen bei der Arbeit mit Paketmanagern wie *npm*<sup>47</sup> und die Arbeit mit Extensions.

Zu den Extensions, die in diesem Projekt angewendet wurden, zählen *Go Live*, der dafür sorgt, dass das aktuelle Projekt nach einem einzelnen Mausklick im Browser ausgeführt wird, *Monokai Pro*, mithilfe deren benutzerdefinierte Farben für die Schlüsselwörter im Programmcode hinterlegt werden können, und *Prettier*, der den Programmcode automatisch formatiert, damit dies der Entwickler nicht tun muss.

---

<sup>44</sup> (Flanagan, 2020, S. 8)

<sup>45</sup> (Flanagan, 2020, S. 427)

<sup>46</sup> (Flanagan, 2020, S. 580)

<sup>47</sup> (Flanagan, 2020, S. 578)

Die Programmierung der LC/NC-Version von LexiCode wurde auf der Plattform *Zoho Creator* durchgeführt. Diese Plattform wurde für die Low-Code-Anwendung ausgewählt, da sie laut der Homepage von Zoho als die „beste Low-Code-Plattform“ bewertet wird<sup>48</sup>.

Zoho Creator bietet die bisher angekündigten Funktionalitäten, die eine typische LC/NC-Plattform kennzeichnen, wie exemplarisch die Möglichkeit, Layouts und Komponenten über Drag-and-Drop zu erzeugen.

In der integrierten Entwicklungsumgebung von Zoho Creator, die als *Page Builder* bekannt ist, können einerseits über Drag-and-Drop Diagramme, Suchfelder, Formulare, Berichte, Buttons und andere gängige Komponenten erstellt werden. Andererseits können über Programmcode Anpassungen vorgenommen werden, die über die Drag-and-Drop Bausteine hinausgehen. Dies ist über HTML-Snippets möglich. Über *Deluge*, die Programmiersprache von Zoho Creator, lässt sich Funktionslogik programmieren.

Der Homepage von Zoho ist zu entnehmen, dass mittels der Plattform Anwendungen für jeden Verwendungszweck erstellt werden können und sie 10-mal schneller in jeder Phase des Applebenszyklus verwaltet werden können als bei anderen Tools<sup>49</sup>. Zoho erklären diese Geschwindigkeit mit der Tatsache, dass für eine Zoho Anwendung kein Setup benötigt wird, eine Bereitstellung mit einem Klick möglich und kein Wartungsaufwand nötig ist.

Die aufgezählten Merkmale einer LC/NC-Plattform, die bei Zoho Creator zugänglich sind, zusammen mit der umfassenden Dokumentation, die Tutorials zu der Plattform und die Geschwindigkeit bei der Entwicklung sind die Gründe, warum Zoho Creator eine gute Auswahl für die Entwicklung der LC/NC-Version von LexiCode ist.

### 3.4 Analyse

Der in der theoretischen Fundierung beschriebene Ansatz des Value-Driven Testing wird bei der Entwicklung von LexiCode angewendet, bei dem der Aufwand, der wahrscheinliche Schaden und aufgrund dieser Metriken der Test-ROI ermittelt werden.

---

<sup>48</sup> (Zoho, 2025)

<sup>49</sup> (Zoho, 2025)

Der Aufwandb wird in Personentagen gemessen. Der wahrscheinliche Schaden wird als die Summe des minimalen und maximalen Schadens geteilt durch 2 erklärt<sup>50</sup>:

$$(\text{Maximaler Schaden} + \text{Minimaler Schaden}) / 2 = \text{Wahrscheinlicher Schaden} \quad (3)$$

Formel 3. Wahrscheinlicher Schaden

Für den maximalen Schaden rechnen Sneed und Jungmayr mit den bereits erwähnten 600 Fehlern, die sich als angenommene Fehlerrate von 6 Fehlern pro 100 Zeilen Programmcode erklären lassen. Um die Schadenshöhe zu ermitteln, rechnen sie mit Sachbearbeitern als Systemnutzern, die einen fixen Stundengehalt erhalten.

Da allerdings vorgesehen ist, dass LexiCode von privaten Nutzern verwendet wird, werden hier fiktive Daten verwendet, die sich auf private Nutzer beziehen. Diese Nutzer sind die Haupt-User-Gruppe und anstatt Stundengehalt für die Arbeit mit der Anwendung wird die *verlorene Arbeitszeit* berücksichtigt. Diese verlorene Arbeitszeit ist die Zeit, die ein lernender Nutzer für sein Hauptberuf einsetzen kann, bei dem er oder sie 20 Euro pro Stunde verdient, anstatt über LexiCode zu lernen. Die Einheit stellt einer Art Opportunitätskosten dar.

Wenn zu einem gegebenen Zeitpunkt 100 Nutzer mit LexiCode arbeiten und ein Systemausfall vorkommt, dann ist es in einem einstündigen Ausfall mit 100 verlorenen Arbeitsstunden zu rechnen – ein maximaler Schaden. Bei einem leichten, halbstündigen Ausfall ist mit 50 verlorenen Arbeitsstunden zu rechnen.

Im besten Fall sind nur zehn Nutzer betroffen und können eine Stunde mit LexiCode nicht arbeiten, die in 10 verlorenen Arbeitsstunden resultiert und einen minimalen Schaden darstellt<sup>51</sup>.

---

<sup>50</sup> (Sneed & Jungmayr, 2011, S. 207)

<sup>51</sup> (Sneed & Jungmayr, 2011, S. 207)

Bei den angenommenen 600 Fehlern ist bei einer Ausfallquote von 10 % mit mindestens 60 Ausfällen und bei 50 % mit 300 Ausfällen zu rechnen. Der maximale Schaden lässt sich folgenderweise ermitteln:

$$\begin{aligned} 60 \times (100 \text{ h a } 20 \text{ €}) &= 120.000 \text{ €} \\ + 300 \times (50 \text{ h a } 20 \text{ €}) &= 300.000 \text{ €} \\ &420.000 \text{ €} \end{aligned}$$

Der minimale Schaden liegt dann bei dem folgenden Ergebnis:

$$\begin{aligned} 60 \times (50 \text{ h a } 20 \text{ €}) &= 60.000 \text{ €} \\ + 300 \times (10 \text{ h a } 20 \text{ €}) &= 60.000 \text{ €} \\ &120.000 \text{ €} \end{aligned}$$

Für den wahrscheinlichen Schaden ergibt sich somit der folgende Wert:

$$(420.000 \text{ €} + 120.000 \text{ €}) / 2 = 270.000 \text{ €}$$

Wenn ein Systemtest mindestens zwei Drittel der existierenden Fehler aufdeckt, kann mit einer Verminderung der Fehleranzahl von 600 auf 200 gerechnet werden, die 400 Fehlern entspricht, die in der Wartung, also in der Betriebsphase nach der Veröffentlichung der Anwendung, nicht behoben werden müssen<sup>52</sup>.

Wenn nach der Verteilung der zu erwartenden Fehler gerechnet wird, ist für die Behebung der Fehler die folgende Anzahl von Personentagen zu erwarten:

$$\begin{aligned} 160 \text{ Anforderungsfehler} \times 2,5 &= 400 \text{ Personentage} \\ 120 \text{ Entwurfsfehler} \times 1,0 &= 120 \text{ Personentage} \\ 120 \text{ Codefehler} \times 0,5 &= 60 \text{ Personentage} \\ \text{Summe: } &580 \text{ Personentage.} \end{aligned}$$

---

<sup>52</sup> (Sneed & Jungmayr, 2011, S. 207)

Bei einem Preis von beispielhaften 800 € pro Entwicklerpersonentag liegen die Fehlerbehebungskosten bei 464.000 €. Zusammen mit den Fehlerfolgekosten von 270.000 € ergibt sich ein Einsparungspotenzial von 734.000 €.

Ein auf 700 Personentagen geschätzter Testaufwand, bei dem die Kosten pro Tester bei z. B. 720 € pro Tag liegen, berechnen sich die Testkosten wie folgt:

$$700 \times 720 = 504.000 \text{ €}$$

Das ROI berechnet sich dann wie folgt:

$$\text{ROI} = (\text{Nutzen} - \text{Kosten}) / \text{Kosten}$$

$$\text{ROI} = (734.000 - 504.000) / 504.000 = 0,45.$$

### 3.5 Fehlerbewertung

Für den Entwicklungsprozess wurden bestimmte Ansätze eingehalten, um die Genauigkeit, Präzision, Wiederholbarkeit und Reproduzierbarkeit der Ergebnisse zu gewährleisten. Um Richtigkeit sicherzustellen, wurden die Anforderungen passend überlegt, sodass sie ein herkömmliches Lehrbuch zum Sprachlernen visuell nachmacht, die durch digitale Funktionen ergänzt, digitalisiert und dematerialisiert wird.

Die Anforderungen wurden durch den Entwickler selbst formuliert, der selbst Erfahrung mit dem Erwerb von zwei Fremdsprachen hat und somit eine genauere Vorstellung davon hat, wie der Lernprozess beim Fremdsprachenerwerb abläuft und wie eine digitale Anwendung diesen am besten abbilden kann.

Die entwickelten Softwarekomponenten wurden isoliert, nachdem sie entwickelt wurden, getestet, um zeitnah sicherzustellen, dass sie wie geplant funktionieren. Dieser Ansatz ähnelt dem *Test-Driven Development (TDD)*, nur mit dem Unterschied, dass die Systemkomponenten manuell getestet werden anstatt den Testprozess zu automatisieren.

Bei der Entwicklung wurde darauf geachtet, dass bereits angelegte Komponenten wie Buttons oder Container wiederverwendet werden, um in dem Programmcode eine Konsistenz zu sichern.

Es wurde selbstverständlich auf Präzision geachtet, damit LexiCode fehlerfrei arbeitet, vor allem wenn es um die Verarbeitung von Daten geht. Um dies sicherzustellen, wurde zum einen beim Anlegen der Komponenten, in denen vom Benutzer Daten angegeben werden, sichergestellt, dass die Komponenten genau mit dem Datentyp arbeiten können, den der Benutzer angeben wird.

Z. B. bei Komponenten auf LexiCode, die Übungsaufgaben darstellen und textuelle Antworten wie Wörter, Satzteile oder ganze Sätze erwarten, wurde im HTML-Code sichergestellt, dass diese Komponenten mit dem Datentyp „Text“ angelegt werden. Entsprechend wurden Funktionen in der Logik, die die Richtigkeit von Übungsantworten zu prüfen haben, so geschrieben, dass diese den Datentyp *String* erwarten.

Zum Zweck der Wiederholbarkeit wurde die Versionierung kontrolliert, indem der aktuelle Programmcode nach jeder Änderung auf der Web-Plattform *GitHub* gespeichert wird. GitHub ermöglicht, dass bei fehlerhaften Entwicklungsarbeiten, die den Programmcode schwer beeinträchtigen, die fehlerfreie Version des Codes als Kopie heruntergeladen und an der Anwendung weitergearbeitet werden kann.

Außerdem hat der Entwickler mittels GitHub die Möglichkeit, von überall und von jedem Gerät auf den Programmcode zuzugreifen und an der Anwendung zu arbeiten. Darüber hinaus wurden die Testszenarien mit den genauen Testparametern dokumentiert, sodass sie unter den gleichen Bedingungen beliebig oft wiederholt werden können.

Damit die Testergebnisse unter den gleichen Bedingungen wieder durchgeführt werden können, wurden Maßnahmen für eine gewisse Reproduzierbarkeit ergriffen. Zum einen wurde die Entwicklungsumgebung dokumentiert, damit sie nachgebildet werden kann. Zum anderen wurden die externen Abhängigkeiten wie das eingesetzte Framework dokumentiert. Es wurden Kriterien für die Erzeugung von unterschiedlichen Testdaten eingesetzt, damit unterschiedliche Testszenarien getestet werden. Die Testdaten wurden dokumentiert.

### **3.6 Technologische Sachzwänge**

Die im letzten Kapitel beschriebenen Ansätze zur Genauigkeit und Präzision sowie die Anwendbarkeit und Verfügbarkeit des Projektes weisen Einschränkungen auf, die zu erwähnen sind, weil sie die Interpretation der Ergebnisse beeinflussen können. Die Einschränkungen liegen größtenteils in der eingesetzten Technologie.

Die eingesetzten HC-Entwicklungswerkzeuge sind ausführlich dokumentiert und können durch die Lektüre diverser Lernmedien wie Bücher, Blog-Artikel, Zeitschriften, Foren, YouTube Tutorials und Video-Kurse auf Online-Plattformen wie Udemy angeeignet werden. In manchen dieser Quellen kann die vollständige Umsetzung kleinerer bis mittelgroßer Projekte verfolgt werden, die große Lernpotenziale für unerfahrene Entwickler bieten.

Da Zoho Creator nicht so lange auf dem Markt wie herkömmliche HC-Entwicklungstechnologien und LC/NC-Entwicklung allgemein eine relativ neue Technologie ist, sind entsprechend weniger Lernmaterialien und Beispielprojekte zu finden. Dies verlangsamt den Lernprozess und erschwert ihn zusätzlich.

Der Entwickler, der sich mit der Entwicklung beider Versionen von LexiCode engagiert hat, kann mehr Erfahrung mit HC- als mit LC/NC-Entwicklung aufweisen. Dies kann dazu geführt haben, dass der Entwickler bei im Zoho Creator gewisse Anforderungen nicht umsetzen konnte, für die es bereits Lösungen oder zumindest Workarounds gibt.

Andererseits kann die Tatsache, dass ein einzelner Entwickler an LexiCode gearbeitet hat, die Richtigkeit negativ beeinflusst haben. In Mehrpersonen-Softwareentwicklungsprojekten können Code Reviews stattfinden, bei denen die Entwickler den Code ihrer Teamkollegen auf Fehler prüfen, damit die Wahrscheinlichkeit der Logikfehler verringert wird.

Der Erfahrungsmangel bei der Entwicklung der LC/NC-Version von LexiCode kann natürlich bei der Entwicklung der HC-Version eine Rolle gespielt haben. In einem Team haben die Entwickler normalerweise unterschiedliche Erfahrungsgrade und die erfahrenen können die weniger erfahrenen bei der Entwicklung unterstützen. Die reine Höhe des Aufwands kann die Qualität der Arbeit des Entwicklers beeinflusst haben, da Menschen tendieren, Fehler zu begehen, wenn sie von zu viel Arbeit überfordert werden.

Der Umfang des Case Study kann die Aussagekraft und Qualität der Forschung beeinflusst haben. Wie bereits erwähnt, ist für die vorliegende Forschung die Entwicklung einer reinen Frontend-Anwendung ohne ein Backend geplant worden.

Eine Anwendung, die über einen Backend verfügt, hat die zweifache Funktionalität einer reinen Frontend-Anwendung und verlangt, dass noch mehr Technologien verwendet werden. Dadurch öffnen sich in so einem Case Study weitere Möglichkeiten für einen Vergleich zweier Ansätze, der die Aussagekraft der Forschung erhöhen kann. Mehr Anpassungsmöglichkeiten für eine HC-Fullstack-Anwendung sagt mehr aus als mehr Anpassungsmöglichkeiten für eine Frontend-HC-Anwendung.

Es ist zu erwähnen, dass im Rahmen dieser Studie die kostenfreie Probeversion von Zoho Creator eingesetzt wurde. Obwohl in den kostenpflichtigen Abonnements mehr Features zu finden sind, beziehen sich viele dieser Features auf Merkmale, deren Umsetzung in der vorliegenden Forschung nicht angestrebt wird, wie z. B. KI-Modelle, Integration mit weiteren Zoho Apps oder Business Intelligence und Analytics<sup>53</sup>. Die Eigenschaften, die in dieser Forschung eigentlich gebraucht werden, sind in dem kostenlosen Abonnement vorhanden.

Ferner ist darauf hinzuweisen, dass das vorliegende Entwicklungsprojekt klarstellen soll, welche Komponenten und Funktionalitäten nur mit den Werkzeugen der Plattform Zoho Creator implementiert werden können, ohne nach Ergänzungswerkzeugen wie andere Programmiersprachen oder Plattformen zu suchen und über diese die Anforderungen umzusetzen. Dies hängt mit dem Ziel zusammen, die Möglichkeiten von Zoho Creator als eigenständiges Werkzeug zu präsentieren.

---

<sup>53</sup> (Zoho, 2025)



## 4 Ergebnisse und Diskussion

Die Ergebnisse der Entwicklung werden in der gleichen Reihenfolge vorgestellt wie die Reihenfolge der Forschungsfragen, die zum Anfang der Studie vorgestellt wurden. Bei der Vorstellung der Ergebnisse werden pro Forschungsfrage im Abwechseln zuerst die Ergebnisse der HC- und danach die Ergebnisse der LC/NC-Anwendung erläutert, bevor es gleichermaßen mit der nächsten Anforderung bzw. fortgesetzt wird.

### 4.1 Seitennavigation

In der HC-Version von LexiCode wurde die lokale Navigation anhand des HTML-Elements `<nav>` angelegt, innerhalb dessen eine ungeordnete Liste (`<ul>`)<sup>54</sup> liegt, in der der Link zu jeder Inhaltsseite als Ankerelement (`<a>`)<sup>55</sup> innerhalb eines Listenelementes (`<li>`)<sup>56</sup> eingebettet ist.

Über das Attribut `href` wird im Code der Link zu der entsprechenden Seite hinterlegt:

```
<nav class="table-of-contents">
  <ul class="content-box">
    <li>
      <a
        class="lehrbuch-kapitel"
        href="../A1/Unterkapitel/a1-1.1-wortschatz.html"
      >1.1 Wortschatz: Häufige Begrüßungen und Ausdrücke</a>
    </li>
```

Abbildung 4. (Link zu einer Inhaltsseite)

---

<sup>54</sup> (Bühler, Schlaich, & Sinner, 2017, S. 15)

<sup>55</sup> (Bühler, Schlaich, & Sinner, 2017, S. 18)

<sup>56</sup> (Bühler, Schlaich, & Sinner, 2017, S. 15)

Die globale Navigation zum Wechseln zwischen den Masken<sup>57</sup> ist Teil der Seitenleistennavigation oder *Sidebar-Navigation*. Außerdem bietet die Seitennavigation Links zu allgemeinen Seiten der Anwendung wie z. B. das Kontaktformular oder das appinterne Forum:

```
<nav class="sidebar-nav">
  <li><a href="../../A1/index-a1.html">A1</a></li>
  <li><a href="../../A2/index-a2.html">A2</a></li>
  <li><a href="../../B1/index-b1.html">B1</a></li>
  <li><a href="../../B2/index-b2.html">B2</a></li>
  <li><a href="../../C2/index-c1.html">C1</a></li>
  <li><a href="../../Allgemein/kontakt.html">Kontakt</a></li>
  <li><a href="../../Allgemein/forum.html">Forum</a></li>
</nav>
```

Abbildung 5. (Sidebar.Menu für globale Navigation)

Hier sind die Links Ankerelemente<sup>58</sup>, die in den Listenelementen des größeren Navigationselements angelegt wurden.

Da in der LC/NC-Version der Anwendung der gleiche Code wie in der HC-Version geschrieben wurde, erscheinen die Menüpunkte zwar als Links auf der UI, aber nachdem sie angeklickt werden, liefert Zoho eine Fehlermeldung, dass die Webseite nicht gefunden werden konnte<sup>59</sup>. Dies führt zu dem Beschluss, dass das *href*<sup>60</sup> Attribut von dem Zoho Creator nicht unterstützt wird, womit die Umsetzung der Seitennavigation erschwert wird.

Die Seitennavigation konnte zumindest über den App-Menüersteller<sup>61</sup> angelegt werden, der die Erzeugung der Seitennavigation per Drag-and-Drop ermöglicht. Da Hyperlinks nur über die Seitennavigation erzeugt werden können, wird sie auf Dauer unübersichtlich, weil sie zu viele Links enthält.

---

<sup>57</sup> (Meyer, 2024, S. 80)

<sup>58</sup> (Bühler, Schlaich, & Sinner, 2017, S. 18)

<sup>59</sup> Siehe Anhang A-1

<sup>60</sup> (Bühler, Schlaich, & Sinner, 2017, S. 19)

<sup>61</sup> Siehe Anhang A-2

Somit stellt die HC-Entwicklung mehr Optionen zur Umsetzung von Navigationskomponenten. Da auf Zoho weniger Optionen vorhanden sind, eignet sich diese Entwicklungsplattform nicht zur Umsetzung von Navigationselementen wie eine Header-Navigation oder ein *Footer*<sup>62</sup>. Ohne solche Komponenten ist die Nützlichkeit einer Webanwendung eingeschränkt.

Die Ergebnisse dieser Anforderung sprechen gegen die Studie von Elshan und Binzer, in der behauptet wird, dass die technischen Einschränkungen der LC/NC-Plattformen durch klare Governance-Strategien und Schulungen überwunden werden können. Obwohl Governance die LC-Nutzung steuert und Schulungen den Citizen Developern mehr Kompetenz bei der Entwicklung mittels LC-Plattformen übermitteln können, bietet keine dieser Maßnahmen eine Alternative für die technischen Einschränkungen dieser Plattformen. In dem Falle der vorliegenden Forschung bieten sie keine Alternative für die Hyperlinks<sup>63</sup>.

Die Möglichkeit, in Zoho eine globale Navigation per Mausklick anzulegen, entspricht den Feststellungen von Krogh und Chun, dass LC eine schnellere Umsetzung von Lösungen ermöglicht. Da die Navigation mit allen Modulen einer Anwendung verbunden sein muss, kann eine beschleunigte Erstellung und Erweiterung der Navigation über LC das Time-To-Market verkürzen und die klassischen Entwickler entlasten, weil die Entwicklung von Citizen Developern übernommen wird<sup>64</sup>.

Allerdings kann so eine Navigation im Fall von Zoho eingeschränkt sein, weil die Plattform *href* nicht unterstützt und somit eine Verlinkung zwischen den Anwendungsmodulen nur über die Seitennavigation von Zoho erfolgen kann. Die Erwartung, dass durch den Einsatz von LC die IT entlastet wird, kann nicht erfüllt werden, weil die Nichtverwendbarkeit von Programmierwerkzeugen wie das *href* den weiteren Einsatz der HC-Entwickler erfordern wird, um das Problem über alternative Wege zu lösen.

Wenn diese Entwickler mehr in der Entwicklung der Benutzeroberfläche involviert sind als erwartet, dann können sie weniger Zeit und Energie in der Entwicklung der Sicherheit der Anwendung und ihre Qualitätskontrolle<sup>65</sup> investieren – ein erwarteter Zugewinn durch LC-Entwicklung, der durch die technischen Einschränkungen von LC-Plattformen nicht vollständig ausgeschöpft werden kann.

---

<sup>62</sup> (Bühler, Schlaich, & Sinner, 2017, S. 51)

<sup>63</sup> (Bühler, Schlaich, & Sinner, 2017, S. 95)

<sup>64</sup> (Krogh & Chun, 2025, S. 216)

<sup>65</sup> (Krogh & Chun, 2025, S. 217)

Krogh und Chun haben Best-Practices und Empfehlungen vorgeschlagen, um das Potenzial von LC-Plattformen zu maximieren. Unter anderem empfehlen sie die Definition einer Citizen-Developer-Strategie und die Bereitstellung geeigneter Schulungen und Plattformen. Sie argumentieren dafür, dass technikaffine Citizen Developer wichtige Ressourcen für die rasante Entwicklung einer Unternehmenslösung sein können<sup>66</sup>.

Obwohl Krogh und Chun das Nutzen nicht spezialisierter Entwickler anerkennen, gestehen sie selbst der Tatsache, dass solche Entwickler technisch versiert sein müssen, damit die LC-Entwicklung einen Mehrwert erzielt. Außerdem bedeutet die Schulung der Citizen Developer durch die herkömmlichen Entwickler, dass die herkömmlichen Entwickler weiterhin für triviale Aufgaben eingesetzt werden und sich nicht in dem gewünschten Umfang auf die Projektbereiche konzentrieren können, die ihre Entwicklererfahrung mehr benötigen wie z. B. die Sicherheit der Anwendung oder die Governance<sup>67</sup>.

Krogh und Chun haben auch Vorteile der LCDPs aufgezählt, von denen einige auch in der vorliegenden Forschung eingetroffen sind. Die Autoren weisen darauf hin, dass mithilfe von LC-Plattformen die Entwicklungszeiten verkürzt und eine digitale, innovative Agilität gefördert werden kann<sup>68</sup>. In einem Szenario, in dem für LexiCode Citizen Developer eingesetzt werden, kann die Navigation mühelos von so einem Entwickler angelegt werden.

Das effektivere Nutzen des Fachwissens von technikaffinen Citizen Developern ist ein weiterer Vorteil. Die Autoren beschreiben, dass Citizen Developer, die auch Fachexperten in ihrem IT-fremden Bereich sind, ihre Prozesse besser kennen und diese effektiver in der LC-Anwendung abbilden können<sup>69</sup>. Eine Lehrkraft an einer Sprachschule kann z. B. die Navigation von LexiCode so anlegen, dass diese Schritt für Schritt den Kursverlauf für jeden Schüler nachbildet, von dem Erwerb der Theorie bis zum Ablegen der Prüfung. Die Programmierung durch solche Key User spart auch Zeit, die normalerweise beim Anforderungsmanagement investiert wird, und Prozessdetails werden dadurch weniger missverstanden<sup>70</sup>.

---

<sup>66</sup> (Krogh & Chun, 2025, S. 223)

<sup>67</sup> (Krogh & Chun, 2025, S. 224)

<sup>68</sup> (Krogh & Chun, 2025, S. 219)

<sup>69</sup> (Krogh & Chun, 2025, S. 215)

<sup>70</sup> (Krogh & Chun, 2025, S. 215)

## 4.2 Manipulation der Seitenelemente

Zum Zweck der Anforderung wurden die Buttons mit dem Titel „Check“ gebaut, die per Mausklick die Benutzerangaben auf Richtigkeit überprüfen können. In der HC-Version von LexiCode wurde für diese Buttons eine JavaScript-Funktion geschrieben, die auf die Input-Felder nach ihrer Klasse zugreift. In dem untenstehenden Bild greift die Funktion auf die Klasse „check-ex1q1“:

```
<p class="p1">  
  1. <input class="ex1q1" type="text" />! Wie geht es dir? (Begrüßung  
    am Morgen)<button class="check-ex1q1 check">Check</button>  
</p>
```

Abbildung 6. (Input-Feld für Übungsantwort gefolgt von einem Prüf-Button)

Im weiteren Ablauf der Funktion wurde eine *if*-Abzweigung gebaut, die das erste Teilaufgabenfeld nach dem Inhalt überprüft. Wenn der Inhalt „Guten Morgen“, eingegeben wird, werden der Hintergrund des Eingabefeldes grün und die Buchstaben weiß gefärbt. Bei einer falschen Antwort werden der Hintergrund rot und die Buchstaben trotzdem weiß gefärbt.

In der Zoho-Version von LexiCode ist die Umsetzung einer DOM-Manipulation nicht möglich. Unter den Optionen von der zohoeigenen Programmiersprache, *Deluge*, sind die Möglichkeiten vorhanden, Formulare, Reports oder Widgets einzubetten, Kontrollstrukturen wie *if*-Abzweigungen zu schreiben, und Daten zu manipulieren<sup>71</sup>. Zwar sind Optionen vorhanden, Strukturen wie Listen, Maps, Webdaten<sup>72</sup> oder die XML zu manipulieren, aber in Bezug auf DOM-Manipulation scheinen keine Möglichkeiten zu finden zu sein<sup>73</sup>.

Die Anwendungskomponenten können somit nur bei einer HC-Anwendung manipuliert werden. Obwohl in Zoho diverse Funktionen implementiert werden können, ist die fehlende Komponenten-Manipulation ein Mangel, der das Nutzererlebnis einschränkt. Komponentenmanipulation kann somit bei LC nur durch mehrere Workarounds umgesetzt werden. Die Erwartung, dass innerhalb des Zoho Creators eine der DOM-Manipulation ähnlichen Technologie verfügbar ist, wurde hier nicht erfüllt.

---

<sup>71</sup> Siehe Anhang A-3

<sup>72</sup> Siehe Anhang A-4

<sup>73</sup> Siehe Anhang A-5

Diese Ergebnisse widersprechen der Forschung von Bärman, weil die Anforderung mittels LC-Entwicklung nicht maßgeschneidert und effektiver<sup>74</sup> sondern überhaupt nicht realisiert werden konnte. Die von ihm vorgeschlagenen Maßnahmen, klare Ziele zu definieren und die Plattform sorgfältig zu steuern, können das Ergebnis nicht optimieren, weil das grundlegende Werkzeug zur Umsetzung der Anforderung fehlt.

Novales und Mancha und Brito zählen mehrere Vorteile von LC-Plattformen auf, von denen aber mehrere im Rahmen der Anforderung an Seitenmanipulation nicht realisiert werden konnten. Zum einen behandeln sie die Entwicklung von Anwendungen mit minimalem Programmieraufwand<sup>75</sup>.

Der Aufwand für die Entwicklung von LexiCode hat sich als nicht minimal erwiesen, da für die Seitenmanipulation keine Lösung innerhalb von Zoho gefunden werden konnte. Dies führt dazu, dass für eine potenzielle Lösung zuerst grundlegend recherchiert werden muss, bevor es dann versucht werden kann, die Lösung zu implementieren.

Weiterhin besprechen Novales und Mancha die beschleunigte Entwicklung und reduzierte Abhängigkeit von spezialisierten IT-Ressourcen<sup>76</sup>. Auch dieser Punkt ist im Falle von LexiCode und bei der LC-Entwicklung nicht eingetroffen, da die Suche nach einer alternativen Lösung entweder eine tiefe Auseinandersetzung mit der technischen Dokumentation von Zoho, oder die Verwendung einer HC-Programmiersprache benötigen wird. Die beiden Möglichkeiten stellen spezialisierte IT-Ressourcen dar.

Die identifizierte Rolle von Citizen Developern als Entwickler eigener Anwendungen steht aufgrund der Ergebnisse der vorliegenden Forschung in Frage und scheint nicht die von Novales und Mancha angesprochene Flexibilität und Innovation zu fördern<sup>77</sup>. Die Herausforderungen bei der Implementierung sind somit erheblicher als vorgestellt und erfordern mehr Maßnahmen, damit die LC-Entwicklung tatsächlich die gewünschten Ergebnisse erbringen kann.

---

<sup>74</sup> (Bärman, 2023, S. 50)

<sup>75</sup> (Novales & Mancha, 2023, S. 221)

<sup>76</sup> (Novales & Mancha, 2023, S. 222)

<sup>77</sup> (Novales & Mancha, 2023, S. 225)

Unter den Empfehlungen, die Novales und Macha formulieren, die sie zur Bewältigung der mit LC-Plattformen verbundenen Schwierigkeiten äußern, schlagen sie vor, dass die technischen Voraussetzungen der erforderlichen Lösungen evaluiert werden, um zu bestimmen, welche LC-Plattform sich am meisten eignet<sup>78</sup>.

Diese Vorgehensweise basiert berechtigterweise darauf, LC-Plattformen zu vergleichen, wird aber für einen vermeidbaren Aufwand sorgen. Der Grund dafür ist die Tatsache, dass über jede überlegte LC-Plattform recherchiert werden muss, ob diese die nötigen Werkzeuge hat, um die Lösung umzusetzen. Darüber hinaus muss festgestellt werden, ob in der Citizen Developer- oder in der HC-Entwicklungsabteilung Entwickler vorhanden sind, die die Kompetenz haben, mit der jeweiligen Plattform zu entwickeln.

Obwohl die Seitnemanipulation zumindest im Rahmen der vorliegenden Forschung nicht umgesetzt werden konnte, ist ihre Umsetzung nicht ausgeschlossen. Da Zoho über eine eigene Programmiersprache verfügt und mit anderen Programmiersprachen oder Entwicklungssysteme kompatibel ist, ist die Implementierung vorstellbar. Dies verlangt, dass die Dokumentation von Zoho und die dazugehörigen Anleitungsvideos studiert und die Kompatibilität von Zoho mit anderen Programmiersprachen oder Entwicklungsumgebungen ausprobiert wird.

Da wie bei anderen LC-Plattformen auch Zoho Creator weiterentwickelt wird, ist es nicht auszuschließen, dass bei einem zukünftigen Update entweder Deluge Funktionen oder sogar Drag-and-Drop-Bausteine die Seitenmanipulation ermöglichen.

---

<sup>78</sup> (Novales & Mancha, 2023, S. 222)

### 4.3 Anpassbares Formular

Für dieses Formular wurde im HTML-Code ein `<form>`-Element angelegt<sup>79</sup>, innerhalb dessen Unterlemente für jedes Feld angelegt wurden, die eine Kombination aus Beschriftungs- und Feldelement darstellen, wie am untenstehenden Beispiel des Eingabefeldes für die E-Mail-Adresse zu entnehmen ist:

```
<div class="p-2 contact-row">
  <label for="email">E-Mail</label>
  <input
    type="text"
    name="email"
    id="email"
    class="contact-input" />
</div>
```

Abbildung 7. (Input-Feld für E-Mail-Adressen)

In den CSS-Einstellungen wurde für das Formular eine blaue Außenkante definiert, die Ecken des Formulars wurden gerundet und Innen- und Außenabstände wurden konfiguriert. Außerdem wurde das Formular mit einer eigendefinierten Breite von 600 Pixel mittig auf der Seite platziert und es wurde ein Schatten definiert, um dem Formular ein 3D-Effekt zu verleihen. Diese CSS-Einstellungen wurden unter der Klasse „contact-form“ definiert:

```
.contact-form {
  border: 1px solid #0ea5e9;
  border-radius: 5px;
  padding: 20px;
  max-width: 600px;
  margin: 20px auto 80px;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
}
```

Abbildung 8. (Eigendefinierte CSS-Klasse für das Kontaktformular)

Im Anhang A-6 ist die finale Version des HC-Formulars ersichtlich und im Anhang A-7 der vollständige HTML-Code des Formulars.

---

<sup>79</sup> (Bühler, Schlaich, & Sinner, 2017, S. 29)



Auf Zoho waren keine Möglichkeiten vorhanden, das Formular visuell anzupassen. Ein erfolgreicher Versuch wurde z. B. auf die Überschrift des Formulars unternommen, um die Überschrift zu zentrieren (Anhang A-9). Desweiteren können der Name des Formularbausteins, der Feldlinkname, die Kopfzeile und der Feldtyp geändert werden (siehe Anhang A-11). Felder können umbenannt, zu Pflichtfeldern gemacht und können PBD zulassen (Siehe Anhang A-12). Präfixe oder Suffixe können definiert werden, die Feldgröße und der Beschreibungstext können definiert werden (siehe Anhang A-13).

Das LC-Formular konnte optisch nicht angepasst werden<sup>80</sup>, ein Umstand, der im Umfang dieses Case Study keine signifikanten Folgen hat, aber bei Anfragen nach optischen Änderungen alternative Lösungen verlangen wird. Trotzdem ermöglicht es das Anlegen und Anpassen von Feldern. Bei beiden Versionen der Anwendung konnte ein Button zum Absenden erzeugt werden.

Das Ergebnis dieser Anforderung entspricht der Forschung von Di Ruscio et al., weil in der vorliegenden Forschung festgestellt wurde, dass LC/NC-Entwicklung die Benutzerfreundlichkeit fördert<sup>81</sup> aber einen Mangel an Anpassungsfähigkeiten aufweist. Wie von ihnen formuliert, ist das MDE und sogar die HC-Entwicklung die zu bevorzugende Entwicklungstechnologie.

Die Erstellung des Kontaktformulars durch reines Auswählen von Eingabefeldern bestätigt die Benutzerfreundlichkeit der LC-Plattformen, die Theam Lim et al. in ihrer Studie thematisiert haben<sup>82</sup>. Die niedrigere Einstiegshürde im Zoho Creator ermöglicht, dass neben dem Kontaktformular noch andere Systemkomponenten von Entwicklern ohne herkömmlichen Programmierkenntnissen entwickelt werden. Die schnelleren ersten Erfolge bei der LC-Entwicklung auf Zoho können dazu führen, dass Anfänger-Entwickler schnelles Vertrauen in ihren eigenen Programmierfähigkeiten sammeln und schneller zu größeren Projekten wechseln.

Die Studie von Gomes und Brito weist einige Punkte auf, die mit den Ergebnissen der Anforderung des vorliegenden Unterkapitels übereinstimmen. Einerseits behaupten die Autoren berechtigterweise, dass durch LC-Plattformen eine schnellere Anwendungsentwicklung möglich ist<sup>83</sup>. Dies hat sich am Beispiel des Kontaktformulars nachweisen lassen, weil das Kontaktformular auf Zoho mit wenig Aufwand und ohne Programmcode angelegt werden konnte.

---

<sup>80</sup> Siehe Anhang A-10

<sup>81</sup> (Di Ruscio, et al., 2022, S. 442)

<sup>82</sup> (Theam Lim, et al., 2024, S. 489)

<sup>83</sup> (Gomes & Brito, 2022, S. 2)

Außerdem hat Zoho, genauso wie Gomes und Brito über LC-Plattformen behaupten, eine Benutzerfreundlichkeit<sup>84</sup>, weil das Kontaktformular über Drag-and-Drop Komponenten gebaut werden konnte, die auch einfach zu finden und zu bedienen sind. Dank dieser Benutzerfreundlichkeit kann die LC-Version von LexiCode auch in einer Konstellation gebaut werden, in der der Entwickler für die vorliegende Studie ein Citizen Developer ist.

Der behauptete Aspekt der Kostensenkung wird im Kapitel 4.7 der vorliegenden Forschung besprochen und weist widersprüchliche Ergebnisse zu den Behauptungen von Gomes und Brito. Die von ihnen behandelte Dimension der Integration mit weiteren Systemen wird, wie früher angedeutet, nicht näher erläutert.

Ein Aspekt der Forschung von Gomes und Brito, mit der die vorliegende Forschung nicht übereinstimmt, betrifft die Anpassungsfähigkeit. Das LC-Kontaktformular konnte anders als das HC-Formular nicht visuell angepasst werden. Diese fehlende Anpassungsfähigkeit war in ähnlicher Form bei den in den vorherigen Unterkapiteln behandelten Anforderungen ebenfalls zu finden und deutet auf eine unvollständige Zuverlässigkeit von LC-Plattformen bei dem Bedarf, über solche Plattformen gebaute Anwendungen an sich ändernden Anforderungen anzupassen.

Die von Gomes und Brito beschriebenen Herausforderungen bei der LC-Entwicklung konnten in der vorliegenden Forschung ebenfalls festgestellt werden. Wie sie angekündigt haben, können LC-Plattformen eine hohe Lernkurve haben<sup>85</sup>, die sich auch am Beispiel des Kontaktformulars verwirklichen kann. Da das Formular über Zoho nicht optisch angepasst konnte, muss die Plattform tiefer studiert werden, um herauszufinden, ob sie weitere Anpassungswerkzeuge besitzt oder ob sie durch eine Integration mit anderen Systemen mehr Anpassungsmöglichkeiten anbieten kann.

Der in der Studie von Gomes und Brito angegebene Anwendungsfall des Unternehmens kann durch den Einsatz von LC-Plattformen zwar durch ein kurzes Time-to-Market profitieren, wird aber langfristig aufgrund der aufgezählten Herausforderungen eine HC-Lösung benötigen.

---

<sup>84</sup> (Gomes & Brito, 2022, S. 2)

<sup>85</sup> (Gomes & Brito, 2022, S. 2)

In der Studie werden auch die positiven Aspekte der LC-Plattformen erwähnt, die auch bei Zoho Creator zutreffen. Zum einen behaupten Gomes und Brito, dass LC-Technologien Programmieranfängern einfach zu erwerben sind<sup>86</sup>. Auch über Zoho können Programmieranfänger das Kontaktformular über wenige Mausklicks erstellen.

Ein weiterer positiver Aspekt, der auch bei LexiCode präsent ist, ist die Fähigkeit, mittels LC-Plattformen schnell auf Branchenänderungen reagieren zu können und diese innerhalb weniger Zeit in der LC-Anwendung zu integrieren. Da Zoho zu vielen Anwendungen eine offene Schnittstelle hat, kann z. B. eine Anbindung zu einer Videokonferenz-Software etabliert werden, um Prüfungen online abzulegen.

#### 4.4 Inzeilige Eingabefelder

Für diese Anforderung wurde eine Liste mit einzelnen Elementen angelegt, in denen Input-Felder vom Typ *text* liegen, weil hier textuelle Daten anzugeben sind:

```
<ul class="bullet-free final-test">
  <li class="txt-align-left final-li">
    a) <input type="text" class="rounded ft2-qa" /> Morgen! (Good morning!)
  </li>
</ul>
```

Abbildung 9. (Inzeilige Eingabefelder mittels HTML)

Das `<ul>`-Element, welches die Listenelemente umschließt, besitzt die vom Entwickler definierte Klasse „bullet-free“, die Stichpunkte entfernt, die Listenelemente in einer untergeordneten Liste in HTML automatisch erhalten<sup>87</sup>:

```
.bullet-free {
  list-style-type: none;
}
```

Abbildung 10. (Eigene Klasse zum Entfernen von Stichpunkten)

---

<sup>86</sup> (Gomes & Brito, 2022, S. 2)

<sup>87</sup> (Bühler, Schlaich, & Sinner, 2017, S. 15)

Der gleiche Code wurde ohne Änderung in einem leeren HTML-Snippet im Zoho Creator übernommen. Das Endprodukt ist vom Aussehen her nahezu identisch mit der HC-Version der Aufgabe.

Da im Unterschied zu Entwicklungsumgebungen für HC-Entwicklung wie bspw. VSC in im Zoho Creator keine getrennten Dokumente oder Ordner für programmcode angelegt werden können, müssen alle für den Abschlusstest relevanten Codezeilen in dem gleichen HTML-Snippet eingefügt werden. Dies wird erreicht, indem der CSS-Code in dem HTML-Tag `<style>`<sup>88</sup> eingefügt und der HTML-Code gleich daruntergeschrieben wird. Diese Vorgehensweise verringert die Übersichtlichkeit bei zunehmender Anzahl an Codezeilen.

Die Anforderung nach inzeiligen Eingabefeldern ist soweit die erste in diesem Case Study, die in beiden Versionen von LexiCode erfüllt werden konnte. Die beiden Entwicklungsansätze bieten daher hinsichtlich Eingabefelder die gleichen Entwicklungsmöglichkeiten und dies spricht für die LC/NC-Entwicklung, da das Anlegen solcher Felder in vielen Anwendungsfällen vorkommt.

Die Ergebnisse dieser Anforderung stimmen mit den Hauptvorstellungen der Studie von Ajimati et al. überein. Anforderungen, die über LC/NC eingehalten werden können, führen zu einer Effizienzsteigerung bei der Entwicklung und können von Citizen Developern realisiert werden, die sog. Demokratisierung der Softwareentwicklung<sup>89</sup>. Obwohl Ajimati et al. die möglichen Qualitätsmängel bei LC/NC-Anwendungen hervorheben, konnte in der vorliegenden Forschung nachgewiesen werden, dass einfachere Anforderungen wie die inzeiligen Eingabefelder mittels LC-Plattformen in der gewünschten Qualität erbracht werden können.

Das Ergebniss dieser Anforderung weist gewisse Parallelen zu der Forschung von Pichidtienthum, Pugsee, und Cooharojananone auf, weil diese darauf abgezielt haben, mittels einen Modul-Generator auf LC-Basis neue Module für das ERP-System Odoo zu generieren<sup>90</sup>. Ähnlich dazu können Seiten auf der LC-Version von LexiCode mittels HTML-Snippets mit vorgefertigtem Programmcode erzeugt werden.

---

<sup>88</sup> (Bühler, Schlaich, & Sinner, 2017, S. 45)

<sup>89</sup> (Ajimati, Carroll, & Maher, 2025, S. 2)

<sup>90</sup> (Pichidtienthum, Pugsee, & Cooharojananone, 2021, S. 529)

In der vorliegenden Forschung wird dadurch ebenfalls Zeit bei der Entwicklung gespart und dadurch besteht die Möglichkeit, dass sich die Entwickler mehr auf die Programmlogik konzentrieren. Dieses Zeitersparnis ist von Vorteil, weil der Lernaufwand für die Programmlogik von Zoho eine unvorhersehbare Zeit in Anspruch nehmen wird.

Im Unterschied zu der vorliegenden Forschung haben Pichidtienthum, Pugsee, und Cooharajanane erforscht, wie schnell und erfolgreich mittels des Code-Generators Odoo Studio Odoo-Nutzer, Nicht-Odoo-Nutzer und Odoo-Entwickler Module erstellen können. Allerdings ist die Modul-Generierung mittels Odoo Studio mit Kosten verbunden<sup>91</sup> und kann in den erzeugten Modulen keinen Code zur Erweiterung der Module generieren. Im Vergleich dazu ist die Entwicklung und Weiterentwicklung mittels HC-Werkzeuge wie VSC kostenfrei.

Als Lösungsweg, bei dem weder über HC programmiert noch ein kostenpflichtiges Modul eingesetzt werden soll, schlagen die Autoren vor, den Module Generator zu verwenden, weil durch diese Module leicht erstellt werden können, die auch gleich verwendet werden können.

Obwohl sie dieses Werkzeug vorschlagen, scheint es nach genauerer Betrachtung nicht für Entwicklung durch Citizen Developer bzw. kein reines LC/NC-Entwicklungswerkzeug zu sein, weil die Autoren selbst angeben, dass ein Entwickler einen Odoo-Server starten und neustarten können muss - eine Fähigkeit, die die Autoren selbst als Entwicklerhintergrund bezeichnen<sup>92</sup>. Diese vorausgesetzten Entwicklerkenntnisse sprechen gegen das Konzept eines LC-Entwicklungstools.

Ein Schritt in dem Prozess der Modulerzeugung in Odoo, der tatsächlich auf purer LC/NC-Basis funktioniert, ist das Erstellen von Views<sup>93</sup>, nachdem Menüs erstellt und auf Models referenziert sind. Bis ein Entwickler aber zu diesem Schritt kommt, hat er die Aufgabe, die Bedienung von Odoo Servern zumindest grundlegend zu beherrschen, die je nach technischer Affinität des Citizen Developers eine unterschiedliche Menge an Zeit in Anspruch nehmen kann.

---

<sup>91</sup> (Pichidtienthum, Pugsee, & Cooharajanane, 2021, S. 530)

<sup>92</sup> (Pichidtienthum, Pugsee, & Cooharajanane, 2021, S. 530)

<sup>93</sup> (Pichidtienthum, Pugsee, & Cooharajanane, 2021, S. 533)

## 4.5 Sitzungsunabhängige Datenspeicherung

Für die Speicherung wurden Input-Felder vom Typ *button* angelegt, die genauso wie herkömmliche Buttons funktionieren und eine Event-Handler-Funktion ausführen:

```
<p>
  1.
  <input type="text" class="rounded score-field g-ex1-q1" id="g-ex1-q1" />
  bin da. ( I )
  <input
    type="button"
    class="btn btn-primary lokal-speichern"
    value="Speichern"
    onclick="speichern()" />
</p>
```

Abbildung 11. (Teilaufgabe zur sitzungsunabhängigen Datenspeicherung)

Die konkrete Funktion namens `speichern()` sorgt dafür, dass der Wert von jedem Feld in der browserseitigen Eigenschaft *localStorage* gespeichert wird:

```
function speichern() {
  let feldWert = document.querySelector(".g-ex1-q1").value;
  let feldWert2 = document.querySelector(".g-ex1-q2").value;
  let feldWert3 = document.querySelector(".g-ex1-q3").value;
  let feldWert4 = document.querySelector(".g-ex1-q4").value;
  localStorage.setItem("g_ex1_q1", feldWert);
  localStorage.setItem("g_ex1_q2", feldWert2);
  localStorage.setItem("g_ex1_q3", feldWert3);
  localStorage.setItem("g_ex1_q4", feldWert4);
}
```

Abbildung 12. (Funktion zur browserseitigen Speicherung von Daten)

LocalStorage dient der fortbestehenden Speicherung von Strings, und macht sie nach Neuladen der Seite für den Benutzer immer noch verfügbar<sup>94</sup>. Damit die Daten abgerufen werden, wurde die separate Funktion „holen()“ geschrieben, die gewährleistet, dass die Daten aus localStorage wieder in die Felder geladen werden:

```
function holen() {  
  let gespeicherterWert = localStorage.getItem("g_ex1_q1");  
  let gespeicherterWert2 = localStorage.getItem("g_ex1_q2");  
  let gespeicherterWert3 = localStorage.getItem("g_ex1_q3");  
  let gespeicherterWert4 = localStorage.getItem("g_ex1_q4");  
  if (  
    gespeicherterWert &&  
    gespeicherterWert2 &&  
    gespeicherterWert3 &&  
    gespeicherterWert4  
  ){  
    document.getElementById("g-ex1-q1").value = gespeicherterWert;  
    document.getElementById("g-ex1-q2").value = gespeicherterWert2;  
    document.getElementById("g-ex1-q3").value = gespeicherterWert3;  
    document.getElementById("g-ex1-q4").value = gespeicherterWert4;  
  }  
}
```

Abbildung 13. (Abrufen zuvor gespeicherter Werte)

Dank der breiten Auswahl an JavaScript Funktionen konnte die Anforderung in der HC-Anwendung umgesetzt werden. Auf Zoho ist eine solche Funktion allerdings nicht vorhanden. Dementsprechend müssen bei der LC-Anwendung andere Methoden zur Speicherung von benutzerdefinierten Daten herangezogen werden, wie z. B. das Aufbauen einer eigenen Datenbank. Da so ein Aufwand bei der Entwicklung einer HC-Anwendung nötig ist und angesichts der Einschränkungen, die bei den bisher vorgestellten Ergebnissen festgestellt wurden, sprechen außer der potenziellen Zeitersparnis wenig weitere Argumente dafür, eine mit eigenem Backend ausgestattete Anwendung über Zoho Creator zu bauen.

---

<sup>94</sup> (Flanagan, 2020, S. 537)

Die Äußerung von Schreiner, dass LC-Plattformen den Einsatz von HC-Entwicklern erübrigen<sup>95</sup>, steht im Widerspruch zu den in diesem Kapitel behandelten Ergebnissen. Da spezifische Anforderungen wie die sitzungsunabhängige Datenspeicherung in vielen Softwareentwicklungsprojekten vorkommen, wird bei einem LC-Projekt eine HC-basierte Zusatzlösung erforderlich sein, die nur von einem HC-Entwickler in die Tat umgesetzt werden kann.

Die Erkenntnisse aus dieser Anforderung ermöglichen, dass Zoho mit anderen LC-Plattformen verglichen wird, genauso wie Sahay, Indamutsa, Di Ruscio und Pierantonio bereits getan haben. Diese haben mehrere LC-Plattformen, einschließlich Zoho, in Bezug auf Interoperabilität, Erweiterbarkeit, Skalierbarkeit und Lernkurve verglichen<sup>96</sup>. Anhand der Erfahrungen, die innerhalb der vorliegenden Forschung mit Zoho gesammelt wurden, kann der Vergleich in manchen Aspekten erweitert werden.

In Bezug auf Erweiterbarkeit haben die Autoren vermerkt, dass die von Ihnen verglichene LC-Tools wie OutSystems, Mendix, MS PowerApp, Google App Maker, Kissflow, Salesforce App Cloud, Appian und Zoho schwer oder teilweise unmöglich zu erweitern sind<sup>97</sup>. Diese Ansicht konnte bisher bestätigt werden, weil in Zoho nicht nur die sitzungsunabhängige Datenspeicherung, sondern auch die Seitenmanipulation, eine erweiterte Seitennavigation und ein anpassbares Formular nicht umgesetzt werden konnten.

Angesichts der Interoperabilität sind die Autoren der Ansicht, dass Artefakte wie Komponenten, architektonische Strukturen und Daten nicht zwischen den unterschiedlichen LC-Plattformen ausgetauscht werden können<sup>98</sup>. Dies ist auch bei Zoho der Fall, denn beim initialen Ausprobieren der allgemeinen Eigenschaften der Plattform festgestellt werden konnte, dass sie mit keinen anderen LC-Plattformen eine offene Schnittstelle teilt.

Die Lernkurve ist laut Sahay, Indamutsa, Di Ruscio und Pierantonio bei den meisten Plattformen nicht intuitiv und besteht aus begrenzten Drag-and-Drop Komponenten, keinen Beispielanwendungen und wenig Dokumentation oder Tutorials in Videoform<sup>99</sup>.

---

<sup>95</sup> (Schreiner & F., 2024, S. 45)

<sup>96</sup> (Sahay, Indamutsa, Di Ruscio, & Pierantonio, 2020, S. 174)

<sup>97</sup> (Sahay, Indamutsa, Di Ruscio, & Pierantonio, 2020, S. 174)

<sup>98</sup> (Sahay, Indamutsa, Di Ruscio, & Pierantonio, 2020, S. 177)

<sup>99</sup> (Sahay, Indamutsa, Di Ruscio, & Pierantonio, 2020, S. 177)



Der Entwickler der vorliegenden Forschung konnte Zoho entnehmen, dass eine gute Zahl von Drag-and-Drop Möglichkeiten, online Dokumentationen und Videotutorials vorhanden sind, von denen z. B. einpotenzieller Zoho Entwickler lernen kann, wie ein CRM-Modul über Zoho gebaut werden kann.

Allerdings müssen diese tiefergehend studiert werden, weil die Programmiersprache von Zoho nicht intuitiv zu bedienen ist und keine Tutorials vorhanden sind, die den Deploymentprozess oder die Erweiterung auf mobile Endgeräte erklären.

Sahay und die Mitautoren sind der Überzeugung, dass LC-Anwendungen einfach zu skalieren sein sollten, aber da es keine offenen Standards für solche Plattformen gibt, kann die Skalierbarkeit solcher Plattformen schwer bewertet, recherchiert oder dazu beigetragen werden<sup>100</sup>. Um die Skalierbarkeit zu beurteilen, muss die Dokumentation auf der Webseite von Zoho berücksichtigt werden.

Laut dem Preis- und Inhaltsvergleich der unterschiedlichen Zoho-Abonnements können im professionellen Modell von Zoho Daten für fünfmal mehr Benutzer verarbeitet werden, es ist dreimal mehr Speicherplatz vorhanden und es können uneingeschränkt viele Zoho Anwendungen gebaut werden im Vergleich zu der Standardversion<sup>101</sup>. Diese erhöhte Leistung ist allerdings mit höheren Kosten verbunden, die je nach Benutzeranzahl der Anwendung noch weiter steigen können.

Obwohl diese Anforderung in der vorliegenden Forschung ebenfalls nicht umgesetzt werden konnte, ist mit weiterer Arbeit mit und Vertiefung in Zoho eine Umsetzung realisierbar. Die Integrationsplattform von Zoho, Zoho Flow, ermöglicht das Erstellen von Workflows durch die Integration mit hunderten von Apps, die auch von der Zahl her ständig wachsen<sup>102</sup>. Durch diese Workflows kann sichergestellt werden, dass die Arbeitsprozesse, die eine Seitenmanipulation erfordern, zumindest in andere Anwendungen stattfinden.

---

<sup>100</sup> (Sahay, Indamutsa, Di Ruscio, & Pierantonio, 2020, S. 178)

<sup>101</sup> (Zoho, 2025)

<sup>102</sup> (Zoho, 2025)

## 4.6 Anforderungen außerhalb der Forschungsfragen

Für diese Anforderungen wurde das CSS-Framework Bootstrap eingesetzt, die z. B. ermöglicht hat, dass die Ecken von Eingabefeldern gerundet werden, ohne eine separate CSS-Klasse zu definieren<sup>103</sup>:

```
<li class="txt-align-left final-li">  
  a) My name is Anna. <input type="text" class="rounded ft1-qa" />  
</li>
```

Abbildung 14. (Bootstrap-Klasse für runde Ecken)

Als weiteres Beispiel besitzt der „Abgeben“ Button die Klasse *btn*<sup>104</sup>, welche dem Button eine Standardhöhe, -breite und Innenabstand vergibt. *btn-primary*<sup>105</sup> sorgt dafür, dass der Button blau gefärbt wird und ein Hover Effekt stattfindet, bei dem sich die Farbe auf dunkelblau ändert, wenn der Benutzer mit dem Mauszeiger über den Button schwebt:

```
<button class="btn btn-primary" id="submit-t1">Abgeben</button>
```

Abbildung 15. (Bootstrap-Klassen für Buttons)

Außerdem können Komponenten über Bootstrap responsive programmiert werden, indem sie in einem umschließenden Element mit der Klasse „container“ gesetzt werden<sup>106</sup>:

```
<div class="container"></div>
```

Abbildung 16. (Responsiver Bootstrap-Container)

---

<sup>103</sup> (Krause, 2020, S. 126)

<sup>104</sup> (Krause, 2020, S. 12)

<sup>105</sup> (Krause, 2020, S. 144)

<sup>106</sup> (Krause, 2020, S. 74)

Die Responsivität stellt sicher, dass z. B. das Kontaktformular bei einer Bildschirmbreite von 550 Pixel, die unter seiner standardmäßigen Breite von 600 Pixel liegt, immer noch ausreichend Abstand zu beiden Bildschirmkanten und genug Innenabstand hat (siehe Anhang A-20). Das Formular behält seine Überschaubarkeit auch bei einer noch geringeren Bildschirmbreite von 350 Pixel (siehe Anhang A-21).

Die Responsivität innerhalb von Zoho wurde anhand des Abschlusstests geprüft und konnte unerwartete Ergebnisse liefern, da es aufgrund eines Erfahrungsmangels dem Entwickler nicht bewusst war, ob Bootstrap mit Zoho kompatibel ist. Dieses unerwartete Ergebnis spricht für die Entwicklung mittels Zoho, weil diese nicht nur durch vorgefertigte Bausteine, sondern durch die Verwendung von Bootstrap schneller durchlaufen kann. Der Abschlusstest wurde bei einer Bildschirmbreite von 550 Pixel getestet und hat sich an dieser angepasst (siehe Anhang A-22).

Ähnlich zu der Forschung von Gialitakis et al. können für LexiCode mittels Bootstrap viele vorgefertigte Systemkomponenten verwendet werden, die zur Prototypenerstellung dienen<sup>107</sup>. Obwohl Bootstrap mit der HC-Entwicklung kompatibel ist, kann in einem alternativen Entwicklungsprojekt, in dem LexiCode in einer kurzen, vorgegebenen Zeit ins Leben gerufen werden muss, über LC entwickelt werden, um die allgemeine Effizienz des Projektes zu steigern.

Aveiro, Freitas, Cunha, Quintal, und Almeida haben ebenfalls zwei Versionen der gleichen Anwendung verglichen, von denen eine als HC- und die andere als LC-Lösung geschrieben wurde. Noch vor dem Vergleich erwähnen die Autoren, dass der Mangel an Anpassungsfähigkeit und Flexibilität einer der Nachteile von LC-Palltformen ist<sup>108</sup>, genau die Ergebnisse, die bei der Umsetzung des Kontaktformulars und der Seitenmanipulation festgestellt wurden.

Die Autoren geben an, dass DISME, die von ihnen gewählte LC-Plattform, bereits ein Modul zur Authentifikation enthält und deswegen bei der Entwicklung über DISME keine Zeit investiert werden muss, um eine eigene Lösung zu bauen<sup>109</sup>. Im Unterschied zu DISME ist bei Zoho Creator kein vorgefertigtes Authentifizierungsmodul vorhanden und eins muss entsprechend programmiert werden. So ein Modul ist mit viel Aufwand verbunden, weil er die Auswahl einer entsprechenden Backend Technologie einschließt und erhöht den Programmieraufwand, der bei einer LC-Plattform wie Zoho geringer sein sollte.

---

<sup>107</sup> (Gialitakis, Tsakalidis, Nousias, & Vergidis, 2024, S. 650)

<sup>108</sup> (Aveiro, Freitas, Cunha, Quintal, & Almeida, 2023, S. 2)

<sup>109</sup> (Aveiro, Freitas, Cunha, Quintal, & Almeida, 2023, S. 7)

Desweiteren werden die zwei Anwendungen von Aveiro und Kollegen in Bezug auf die Anzahl von Codezeilen von einigen Komponenten innerhalb des Systems verglichen. Es wurden z. B. 6025 Codezeilen für die Client-Seite der HC-Anwendung geschrieben, während es für die LC-Anwendung 501 Codezeilen waren, die eine 91,7 % Reduktion in Komplexität nachweisen<sup>110</sup>.

Ohne zu viel über die Gesamtcodezeilenanzahl der beiden Versionen von LexiCode zu offenbaren, weil diese im nächsten Unterkapitel näher behandelt werden, wird hier erwähnt, dass sich die Anzahl der LC-Codezeilen kaum von der Anzahl an HC-Codezeilen unterscheidet. Dieser Befund stellt die Frage, inwiefern Zoho überhaupt eine LC-Plattform ist.

In der Studie wird berichtet, dass die LC-Version der Anwendung einfacher anzupassen ist, weil z. B. das Layout der Formulare zu jeder Zeit angepasst werden konnte, ohne den Betrieb des Systems zu beeinflussen<sup>111</sup>. Genau über die Zoho Formulare wurde in einem der vorherigen Kapitel herausgefunden, dass sich diese auf Zoho von dem Layout her nicht anpassen lassen.

Aveiro und seine Mitautoren geben weiter an, dass der Erfolg hinter dem LC-Ansatz ihrer Forschung mit ihrer Auswahl der LC-Plattform zusammenhängt. Da DISME eine breite Auswahl an vorgefertigten Komponenten bietet, können sich die Entwickler auf die Applikationslogik und das User Experience konzentrieren, anstatt herkömmlich zu programmieren<sup>112</sup>. Da diese Freiheit mit Zoho nicht erreicht werden konnte, kann festgestellt werden, dass DISME eine reinere LC-Plattform als Zoho ist.

## 4.7 Ermittlung der Testkosten

Die Unterschiede in der Anzahl der Codezeilen, die für die Entwicklung der HC- und der LC/NC-Version von LexiCode geschrieben wurden, führen dazu, dass beide Versionen der Anwendung unterschiedlich hohe Testkosten in Anspruch nehmen werden. Für die HC-Version der Anwendung wurden 1435 Zeilen Code gezählt, die mit dem Faktor 0,05 multipliziert wurden, um 71,75 Solltestfälle zu erhalten.

---

<sup>110</sup> (Aveiro, Freitas, Cunha, Quintal, & Almeida, 2023, S. 9)

<sup>111</sup> (Aveiro, Freitas, Cunha, Quintal, & Almeida, 2023, S. 9)

<sup>112</sup> (Gialitakis, Tsakalidis, Nousias, & Vergidis, 2024, S. 651)

Der Faktor stellt das Verhältnis der 5000 Solltestfälle, die laut Sneed und Jungmayr pro 100.000 Zeilen Code anfallen, da  $5000/100.000 = 0,05$  ergibt. Die im Anhang B-1 ausgewiesenen Werte der Aufwandsberechnung für die HC-Version von LexiCode ergeben den folgenden Aufwand:

$$\text{Aufwand} = 1 \times [(71,75) \times 0,67/4 + (4 \times (1 - 0,4)1,03 \times (0,5/0,45)))]$$

$$\text{Aufwand} = 1 \times ((48,07/6,4)1,03 \times 1,11)$$

$$\text{Aufwand} = 1 \times (81,03 \times 1,11)$$

$$\text{Aufwand} \approx 8,51 \text{ Personentage}$$

Für die Schadensermittlung wird von 8,61 Fehlern nach der Auslieferung der Anwendung ausgegangen. Für die Ermittlung dieser Zahl wird angenommen, dass pro 100 Zeilen Code 6 Fehler auftreten werden, und es wird die Gesamtzahl von 1435 Codezeilen der HC-Version von LexiCode berücksichtigt. Die Schadensermittlung liefert als Ergebnis einen wahrscheinlichen Schaden von 3.871 €, wenn bei einer Ausfallquote von 10 % 0,861 Fehler ( $8,61 \times 0,1 = 0,861$ ) und bei einer Ausfallquote von 50% 4,3 Fehler auftreten ( $8,61 \times 0,5 = 4,3$ ).

Der Entwicklerpersonentag wird anders als bei Sneed und Jungmayr auf 200 € festgelegt und führt somit zu Fehlerbehebungskosten i. H. v. 1.722 € ( $200 \times 8,61 = 1.722$ ). Wenn die Fehlerfolgekosten von 3.871 € darauf gerechnet werden, entsteht durch das umfangreiche Testen ein Einsparungspotenzial von 5.593 €. Der auf 8,51 Personentage geschätzte Testaufwand ergibt zusammen mit den Testkosten von 150 € pro Tag die folgenden Testkosten:

$$8,61 \times 150 \text{ €} = 1.291,5 \text{ €}.$$

Das ROI für die HC-Version von LexiCode berechnet sich wie folgt:

$$\text{ROI} = (5.593 - 1.291,5) / 1.291,5 = 3,33.$$

Für die LC/NC-Version von LexiCode wurden 1487 Zeilen Code gezählt und ergeben mit dem Faktor 0,05 eine Anzahl von 74,35 Solltestfällen ( $1487 \times 0,05 = 74,35$ ). Im Anhang B-3 wurden die Werte für die Aufwandsabrechnung ausgewiesen und diese führen zu der folgenden Schätzung:

$$\text{Aufwand} = 1 \times [(74,35) \times 0,67/4 + (4 \times (1 - 0,4)1,03 \times (0,5/0,45))]$$

$$\text{Aufwand} = 1 \times ((49,81/6,4)1,03 \times 1,11)$$

$$\text{Aufwand} = 1 \times (81,03 \times 1,11)$$

$$\text{Aufwand} \approx 8,51 \text{ Personentage}$$

Für die LC-Anwendung wird davon ausgegangen, dass 8,9 Fehler nach der Veröffentlichung im Code zu finden sein werden. Die Anzahl wurde nach dem gleichen Ansatz wie bei der HC-Anwendung ermittelt. Der wahrscheinliche Schaden für die LC-Anwendung liegt bei 4.005 €, wenn bei einer Ausfallquote von 10% 0,89 Fehler ( $8,9 \times 0,1 = 0,89$ ) und bei einer Ausfallquote von 50% 4,45 ( $8,9 \times 0,5 = 4,45$ ) Fehler auftreten.

Hier gilt der gleiche Entwicklerpersonentag von 200 € und ergibt mit der erwarteten Fehleranzahl Fehlerbehebungskosten i. H. v. 1.780 € ( $200 \times 8,9 = 1.780$ ). Wenn diese mit dem wahrscheinlichen Schaden summiert werden, entsteht ein Einsparungspotenzial von 5.785 €. Der Testaufwand von 8,9 Personentagen ergibt mit den gleichen Testkosten von 150 € pro Person/pro Tag die folgenden Testkosten:

$$8,9 \times 150 \text{ €} = 1.335 \text{ €}.$$

Das ROI lässt sich wie folgt berechnen:

$$\text{ROI} = (5.785 - 1.335) / 1.335 = 3,33$$

Einerseits hat die Schätzung nach dem Value-Driven Testing festgestellt, dass ausführliches Testen in der Entwicklungsphase von LexiCode wirtschaftlich berechtigt ist, weil bei beiden Versionen der Anwendung ein positiver ROI kalkuliert wurde. Außerdem ist dieser ROI höher als der Wert, den Sneed und Jungmayr in ihrem Beitrag berechnet haben.

Allerdings ist zu beachten, dass sie in ihrem Beitrag über eine Konstellation sprechen, in der eine viel umfangreichere Anwendung mit deutlich mehr Codezeilen zu testen ist, die u. a. mehr Solltestfälle und höhere Testkosten verursacht.

Andererseits hat die Schätzung für beide Versionen von LexiCode den gleichen ROI bekommen, der ein Widerspruch zu den Erwartungen darstellt, weil es davon ausgegangen wurde, dass der ROI für die LC/NC-Version einen höheren ROI haben wird. Diese Erwartung wurde nicht erfüllt und ist auf die Anzahl der Codezeilen der beiden Versionen der Anwendung zurückzuführen. Dies stellt ein Paradoxon dar, weil vor der Entwicklung erwartet wurde, dass die LC/NC-Version von LexiCode aus insgesamt weniger Codezeilen bestehen wird, daher der Begriff „Low-Code/No-Code“.

Die fehlende Dokumenten- und Verzeichnisstruktur, die bei der Vorstellung der inzeiligen Eingabefeldern angesprochen wurde, führt dazu, dass auf der Zoho-Anwendung viel Coderedundanz entstanden ist, weil sich wiederholender Code, wie z. B. die CSS-Einstellungen, nicht in einer separaten Datei abgelegt werden kann, aus der die Einstellungen in dem jeweiligen HTML-Snippet ausgelesen werden können.

Manche Komponenten, wie z. B. das Kontaktformular oder die Sidebar-Navigation wurden komplett über NC programmiert. Die Sidebar-Navigation muss auf jeder Seite der Anwendung vorhanden sein und ihre Umsetzung durch vorgefertigte Bausteine hat mehrere Codezeilen gespart. Die LC-Version von LexiCode ist trotzdem als Codebasis umfangreicher.

Wenn eine zukünftige Version von Zoho Creator die Dokumentenstruktur einer Entwicklungsumgebung für HC-Entwicklung unterstützt, dann können diese Coderedundanzen vermieden werden und das vorliegende Case Study würde, wie erwartet, eine geringere Codezeilenanzahl für die LC/NC-Version von LexiCode aufweisen.

In der aktuellen Version von Zoho Creator wird die Anzahl der Codezeilen zwischen den beiden Anwendungsversionen möglicherweise größer und die Codebasis der HC-Anwendung doch wie erwartet größer, wenn noch weitere Komponenten und ein Backend Teil der Anforderungen werden.

Liu et al. (2023) vertreten die gleiche Ansicht wie Sneed und Jungmayr, weil sie festgestellt haben, dass 60 % der Fehler einer Anwendung in der Design- und Spezifikationsphase auftreten<sup>113</sup>. Diese Feststellung unterstützt den Ansatz des Value-Driven Testing, weil es für ein Softwareentwicklungsprojekt kosteneffektiver ist, durch umfangreiches Testen in der Entwicklungsphase viele Fehler aufzudecken und diese zu beheben, bevor das System veröffentlicht wird.

## **4.8 Bestätigung der These**

Die zu Anfang dieser schriftlichen Ausarbeitung vorgestellte These lässt sich nach der Darlegung der Ergebnisse und die Diskussion über ihre Bedeutung sowie der Vergleich der Ergebnisse mit den Erwartungen bestätigen. Im Unterschied zu der LC/NC-Version von LexiCode konnten die Anforderungen auf der HC-Version von LexiCode wie geplant umgesetzt werden. Das gewünschte minimalistische Design der Anwendungskomponenten, die Nutzerinteraktionen und ihre Ergebnisse konnten nah zu den Anforderungen ins Leben gerufen werden.

Dadurch, dass sich die geplanten Komponenten in der HC-Version der Anwendung wie geplant programmieren haben lassen, ist kein zusätzlicher Aufwand für die Suche nach alternativen Werkzeugen wie andere Funktionen, Frameworks, Programmiersprachen, Entwicklungsumgebungen oder alternative Komponenteideen entstanden. Die Implementierung der LC/NC-Version dagegen hat einige unerwartete Hürden bereitgestellt.

## **4.9 Die Ergebnisse auf dem magischen Viereck abbilden**

Die zum Anfang dieser Forschung vorgestellte Studie von Baumgarten et al. hat auf dem sog. magischen Viereck die Ergebnisse mehrerer weiterer Studien zum Thema LC/NC-Entwicklung abgebildet, die entweder eigene Praxiserfahrungen aus eigenen Projekten mitteilen oder meinungsbasierte Beiträge zum Thema LC/NC-Entwicklung geleistet haben. Die Ergebnisse der vorliegenden Studie zeigen einen Zugewinn in Bezug auf Qualität, weil sich die Anforderungen so wie gewünscht haben umsetzen lassen.

---

<sup>113</sup> (Liu, Jiang, Guo, Chen, & Qiao, 2024, S. 695)



Außerdem erzielte das vorliegende Projekt eine Wertesteigerung in der Dimension der Flexibilität, weil manche Anforderungen über einfachere und kreativere Wege als die gängigen Lösungen implementiert werden konnten, wie die sitzungsunabhängige Speicherung der Daten, die sich über die localStorage Eigenschaft des Browsers speichern lassen, anstatt eine Anbindung an eine Datenbank zu benötigen.

Somit erhält die HC-Version von LexiCode den Faktor +1 auf den Dimensionen der Qualität und der Flexibilität auf dem Viereck:

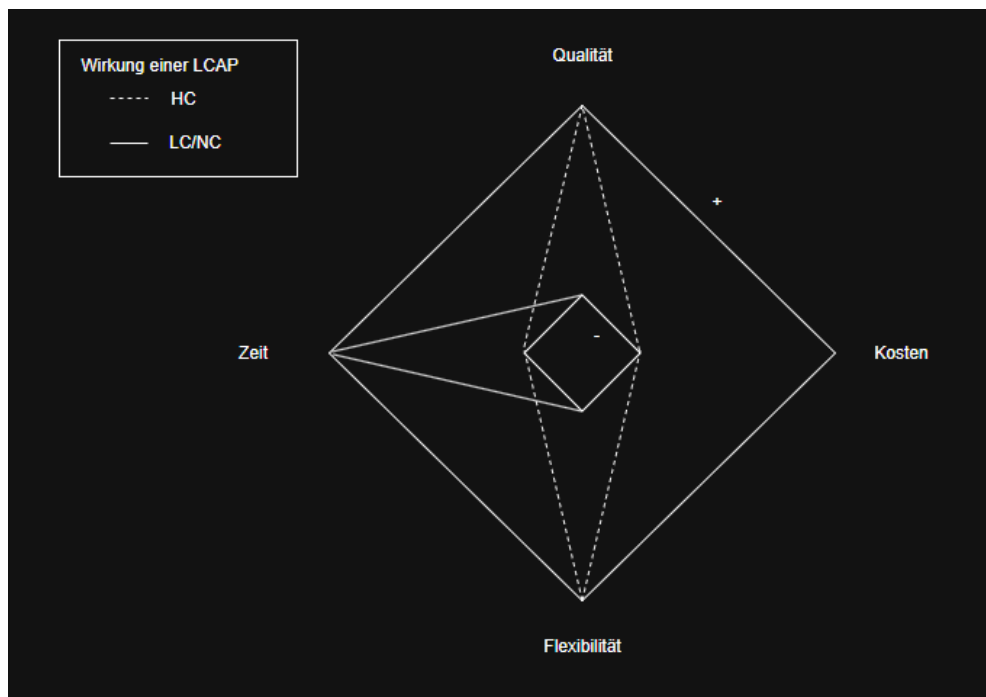


Abbildung 17. (Magisches Viereck mit den Ergebnissen von LexiCode)

#### 4.10 Zusammenfassung

Der Studie ist zu entnehmen, inwiefern die HC- im Vergleich zu der LC/NC-Entwicklung einen vereinfachten Entwicklungsprozess und eine genauere Umsetzung der Systemanforderung zulässt. Die zum Anfang der vorliegenden Forschung aufgezählten Forschungsfragen lassen sich wie folgt beantworten:

**Anlegen von Hyperlinks über der globalen Navigation hinaus, die den Benutzer von einer Seite innerhalb der Anwendung zu einer anderen verleiten sollen.**

So eine Funktionalität lässt sich über die HC-Anwendung, aber nicht über die LC/NC-Anwendung umsetzen. Obwohl Hyperlinks eine der Grundeigenschaften von HTML sind, werden sie vom Zoho Creator nicht unterstützt.

### **Eine Manipulation der Seitenelemente und –komponenten, um mehr Nutzereffekte zu erreichen**

Über die HC-Entwicklung und noch genauer gesagt, die DOM-Manipulation von JavaScript, kann die Grundlage für solche Effekte programmiert werden. Die Standardfunktionen von Zoho Creator unterstützen keine ähnlichen Eigenschaften und lassen sich eventuell nur über Workarounds realisieren.

### **Breitere Auswahl an Anpassungsmöglichkeiten für Webformulare**

Die HC-Entwicklung bietet mehr Möglichkeiten zur Gestaltung eines Webformulars mit Eigenschaften wie Randfarbe, Innen- und Außenabstände und Platzierung auf der Seite. Die Option des Zoho Creators, über vorgefertigte Bausteine Formulare zu erstellen, bietet weniger Änderungsmöglichkeiten.

### **Einbau inzeiliger Eingabefelder und deren Anpassung**

Dies ist eine der Anforderungen, bei denen HC- und LC/NC-Entwicklung innerhalb dieses Case Study gleichgestellt sind. Diese Art von Feldern ergibt sich aus einer Kombination zwischen HTML und CSS und der Code lässt sich über beide Entwicklungsansätze ausführen.

### **Sitzungsunabhängige Speicherung benutzerdefinierter Daten**

Bei dieser Anforderung ist eine Lösung nur über die HC-Version von LexiCode möglich. Die Standardfunktionen von dem Zoho Creator können diese Anforderungen nicht verwirklichen lassen und erfordern den Einsatz von Workarounds.

## 5 Schlussfolgerung

Die HC-Entwicklung erweist sich im Rahmen der vorliegenden Forschung als mehr vorteilhaft für einen Entwicklungsprozess. Diese Aussage basiert darauf, dass innerhalb des vorliegenden Entwicklungsprojektes die vorgesehene Anwendung, die über herkömmliche Programmiersprachen programmiert wurde, über die eigenen Entwicklungswerkzeuge dieser Programmiersprachen programmiert werden konnte, ohne auf zusätzliche Werkzeuge greifen zu müssen. Obwohl es festgestellt wurde, dass bei der Gestaltung des Frontends die LC/NC-Entwicklung mit vielen Instrumenten der HC-Entwicklung arbeiten kann, haben manche teilweise einfache Instrumente gefehlt.

Darüber hinaus hat die LC-Entwicklung und Zoho Creator Einschränkungen im Bereich der Funktionslogik ergeben, da die Plattform nicht genug eigene Funktionen für die Implementierung gängiger Anforderungen anzubieten scheint. Dadurch verlängert und erschwert sich der Entwicklungsprozess, weil Alternativwerkzeuge und -lösungen gesucht werden müssen, um die Anforderungen einzuhalten. Außerdem hat die LC/NC-Entwicklung im Rahmen der vorliegenden Forschung sein Hauptvorteil des Codezeilensparnisses nicht einhalten können, weil die LC-Version von LexiCode aus mehr Codezeilen als die HC-Version besteht.

### 5.1 Kritische Reflexion

Die bei der Umsetzung der LC-Version von LexiCode festgestellten Funktionsmängel sind, laut den Ergebnissen der Forschung, für Zoho Creator spezifisch. Da die Auswahl der LC-Plattform auf der eigenen Beschreibung von Zoho basiert, war die Auswahl der Plattform nicht auf die für das vorliegende Entwicklungsprojekt relevanten Platfformeigenschaften fokussiert.

Bei der Interpretation der Ergebnisse wurde zwar erklärt, welche der Anforderungen nicht umgesetzt werden konnten, und es wurden keine alternativen Lösungen oder zumindest Ansätze vorgestellt, die zu einer Lösung führen können.

Obwohl dies getan wurde, weil das im Forschungsdesign bereits so festgelegt war, ist es für die Plausibilität und Effektivität der LC-Entwicklung trotzdem ein Vorteil, Alternativlösungen für technische Einschränkungen zumindest in der Theorie zu erwähnen. Wenn eine Anforderung über Zoho Creator in Kombination mit anderen Werkzeugen implementiert worden ist, hat Zoho Creator trotzdem einen Beitrag zum finalen Ergebnis und hat die Implementierung ermöglicht.

Bezüglich des LC-Entwicklungsansatzes ist zu erwähnen, dass die beiden Versionen von LexiCode ungefähr die gleiche Menge an Code beinhalten, weil für die LC-Anwendung viel Code überhaupt verwendet wurde. Da der Programmcode für das Aussehen der Komponenten, der für die HC-Version von LexiCode geschrieben wurde, unverändert für die LC-Version übernommen wurde, hat dies dazu geführt, dass beide Versionen der Anwendung eine nahezu gleiche Anzahl an Codezeilen haben.

Obwohl der Ansatz des Value-Driven Testing eine ungefähre Schätzung der Wirtschaftlichkeit hinter dem systematischen Testen von LexiCode vor der Veröffentlichung liefert, stellt sich die Frage, ob er für LexiCode ein geeigneter Analyseansatz ist, da er für die Kalkulation der Testwirtschaftlichkeit betrieblich eingesetzter Software verwendet wird, bei der von einem wirklichen wirtschaftlichen Schaden gesprochen werden kann. Dies liegt daran, dass vergütete Sachbearbeiter mit der Software in der Ausfallzeit nicht arbeiten können und gegen diese vergütete Arbeitszeit kein betriebswirtschaftlicher Wertezuwachs generiert wird.

## **5.2 Methodische Reflexion**

Obwohl die Ergebnisse der vorliegenden Forschung von einem Entwickler bewertet wurden, der sowohl mit HC- als auch mit LC-Technologien Erfahrung hat, ist die Bewertung trotzdem subjektiv und von einer einzigen Person durchgeführt worden. Die Ergebnisse wurden von keinen weiteren Personen bewertet, unabhängig von der Entwicklererfahrung.

Neben dem ROI, dem wahrscheinlichen Schaden und der Aufwandsschätzung wurden keine weiteren objektiven Metriken wie z. B. Entwicklungszeit für die Bewertung herangezogen. Mit den drei objektiven Metriken, die tatsächlich verwendet wurden, wurde nur der Umfang der Codebasis und nicht die einzelnen Anforderungen bewertet.

Eine weiterführende Validierung könnte durch einen klassischen A/B-Test stattfinden, in dem Benutzer die HC- und die LC-Version von LexiCode entweder nur auf Basis der UI vergleichen oder Zugang zu beiden Systemen erhalten und diese testen.

### 5.3 Empfehlungen für die künftige Forschung

Damit unterschiedliche Ergebnisse als die in dieser Forschung erzielt werden können, muss sie unter unterschiedlichen Bedingungen ausgeführt werden. Zum einen muss bei einem vergleichendem Case Study wie das vorliegende die auszuwählende LC/NC-Plattform vor der Auswahl anhand der Dokumentation auf der Seite des Anbieters sowie weitere verfügbare Materialien wie Online-Tutorials grundlegend studiert werden und entwicklerseitig ist es zu analysieren, ob mit der jeweiligen LC-Plattform bereits ähnliche Lösungen umgesetzt worden sind.

Da einige der Anforderungen allein über Zoho nicht programmiert werden konnten, können über die Integrationsmöglichkeiten mit anderen Plattformen Entwicklungsumgebungen wie VSC oder *Eclipse* an die Zoho Anwendung angebunden werden.

Damit in der LC/NC-Version der Anwendung die Anzahl der Codezeilen geringer als in der HC-Anwendung ist, müssen bei einem zukünftigen Case Study die vorgefertigten Bausteine des Zoho Creators bzw., wenn eine andere Plattform als Zoho ausgewählt wird, die vorgefertigten Bausteine der ausgewählten LC-Plattform verwendet werden. Im Falle von Zoho müssen entsprechend die Vorlagen für Formulare, Seiten und Berichte eingesetzt werden, weil diese sogar über Mausklick anstatt über Drag-and-Drop erstellt werden können.

### 5.4 Ausblick

Trotz den genannten technischen Einschränkungen hat die LC/NC-Entwicklung Potenzial im Bereich der schnelleren und einfacheren Entwicklung von kleineren Anwendungen, die mit der Zeit nicht skaliert werden müssen.

Die Zeitersparnis, die LC-Plattformen bei der Entwicklung bieten, äußern sich als vorgefertigte Bausteine, die die optische Entwicklung der Systemkomponenten beschleunigen. Die technischen Einschränkungen in Form von fehlenden Funktionen können bedeuten, dass es auf LC-Plattformen in dem aktuellen technologischen Stand einfach noch keine vorgefertigten Bausteine dafür entwickelt worden sind.

Da Bausteine für die UI-Komponenten und sogar Bausteine für allgemeine, einfachere Funktionen vorhanden sind, ist es zu erwarten, dass in der Zukunft Bausteine für komplexe Funktionen existieren, bei denen jeder Teilvorgang der Funktion ebenfalls per Mausklick oder Drag-and-Drop erzeugt werden kann.

Eine zukünftige Entwicklungslandschaft, in der komplexere, größere Systeme über LC-Plattformen geschrieben werden, wird den Bedarf an HC-Entwicklern reduzieren und dazu führen, dass diese sich mit anderen Aufgabenbereichen beschäftigen müssen.

## Referenzen

- 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*.
- Atkinson, R. (1999). Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management*, pp. 337-342.
- Aveiro, D., Freitas, V., Cunha, E., Quintal, F., & Almeida, Y. (2023). Traditional vs. low-code development: comparing needed effort and system complexity in the NexusBRaNT experiment. *2023 IEEE 25th Conference on Business Informatics (CBI) Business Informatics (CBI), 2023 IEEE 25th Conference on*, pp. 1-10.
- Bärmann, F. (2023). ERP: The next big thing: Sind Lösungen auf Low-Code-Basis die Zukunft? *IT Management*, pp. 50-53.
- Baumgarten, C., Endl, R., & Stich, S. (2024). Professionelle Softwareentwicklung mit Low Code optimieren – eine Fallstudie. *HMD Praxis der Wirtschaftsinformatik*(5), pp. 1213-1234.
- Boehm, B. (1981). *Software Engineering Economics*. Englewood Cliffs: Prentice Hall.
- Böhler, T. (2023). Software einfacher, flexibler und leichtfüßiger entwickeln. *Produktion*(12).
- Brandt-Pook, H. (2020). *Softwareentwicklung kompakt und verständlich : Wie Softwaresysteme entstehen*. Wiesbaden: Springer.
- Bühler, P., Schlaich, P., & Sinner, D. (2017). *HTML und CSS3 : Semantik - Design - Responsive Layouts*. Berlin: Springer.
- Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., & Wimmer, M. (2022). Low-code development and model-driven engineering: Two sides of the same coin? *Software & Systems Modeling*, pp. 437-446.
- Elshan, E., & Binzer, B. (2024). Mehr als ein Trend?: Wie Low-Code die digitale Transformation unterstützt. *HMD Praxis der Wirtschaftsinformatik*(5), pp. 1070-1087.
- Flanagan, D. (2020). *JavaScript: The Definitive Guide : Master the World's Most-Used Programming Language*. O'Reilly.
- Gialitakis, M.-K., Tsakalidis, G., Nousias, N., & Vergidis, K. (2024). Rapid prototyping of process-driven applications using low-code development platforms: A case study from the Greek public sector. *2024 International Conference Automatics and Informatics (ICAI), Automatics and Informatics (ICAI), 2024 International Conference*, pp. 650-656.
- Gomes, P. M., & Brito, M. A. (2022). Low-Code Development Platforms: A Descriptive Study. *2022 17th Iberian Conference on Information Systems and Technologies (CISTI), Information Systems and Technologies (CISTI), 2022 17th Iberian Conference on*.
- Hensen, U. (2023). Maßgeschnittene Software mittels Low Code. *Factory Innovation*(5), pp. 32-36.
- Krause, J. (2020). *Introducing Bootstrap 4*. New York: Apress.
- Krogh, E., & Chun, M. W. (2025). Enlisting Citizen Developers to Deliver Digital Business Value With Generative AI & Low-Code Development Platforms. *Journal of Applied Business & Economics*, pp. 213-225.

- Liu, D., Jiang, H., Guo, S., Chen, Y., & Qiao, L. (2024). Bugs, What's Wrong With Low-Code Development Platforms? An Empirical Study of Low-Code Development Platform. *IEEE Transactions on Reliability, Reliability, IEEE Transactions on*(1), pp. 695-709.
- Meyer, A. (2024). *Softwareentwicklung : agile Methoden, moderne Softwarearchitektur, beliebte Programmierwerkzeuge*. München: De Gruyter.
- Novales, A., & Mancha, R. (2023). Fueling Digital Transformation with Citizen Developers and Low-Code Development. *MIS Quarterly Executive*(3), pp. 221-234.  
doi:10.17705/2msqe.00083
- Phalake, V. S., Joshi, S. D., Rade, K. A., Phalake, V. S., & Molawade, M. (2024). Optimization for achieving sustainability in low code development platform. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, pp. 5717-5724. doi:10.1007/s12008-023-01338-0
- Pichidienthum, S., Pugsee, P., & Cooharajanane, N. (2021). Developing Module Generation for Odoo Using Concept of Low-Code Development Platform and Automation Systems. *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA), Industrial Engineering and Applications (ICIEA), 2021 IEEE 8th International Conference on, 2021*423, pp. 529-533.
- Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Software Engineering and Advanced Applications (SEAA), 2020 46th Euromicro Conference on*, pp. 171-178.
- Schreiner, C., & F., K. (2024). Entwicklungszyklen halbieren mit Low Code. *Digital Engineering*(7), pp. 44-45.
- Siroker, D. (2013). *A/B testing : the most powerful way to turn clicks into customers*. Hoboken: Wiley.
- Sneed, H. M., & Jungmayr, S. (2011). Mehr Testwirtschaftlichkeit durch Value-Driven-Testing. *Informatik-Spektrum: Organ der Gesellschaft für Informatik e.V. und mit ihr assoziierter Organisationen*(2), pp. 192-209.
- Theam Lim, J., Kang Lee, C., Heng Lim, E., Voon Wong, P., Tak Yew, K., & Yee Ooi, C. (2024). Preliminary Study on the Accessibility and Learning Experience of Low-Code Development Platforms. *2024 5th International Conference on Artificial Intelligence and Data Sciences (AiDAS) Artificial Intelligence and Data Sciences (AiDAS), 2024 5th International Conference on*, pp. 486-491.
- Zoho. (2025, April 04). Retrieved from <https://www.zoho.com/de/creator/pricing-comparison.html>
- Zoho. (2025, April 04). Retrieved from <https://www.zoho.com/creator/help/#documentation>
- Zoho. (2025, April 26). Retrieved from <https://www.zoho.com/de/flow/?src=zGlobalRelatedProducts>





## Appendix A. Abbildungen

### A-1 Leere Zoho-Seite nach Betätigen eines HTML-Links

---



## A-2 Seitennavigation über den App-Menüersteller

App-Menüersteller ⓘ

Komponenten

Elemente

Suche

Benutzerkomponenten

Aufsatz

Antworten A2

Aufsatz A1

Auswählen

Symboleinstellungen

Kapitel 1: Begrüßungen und Vorstellu...

1.1 Wortschatz: Häufige Begr...

1.2 Grammatik: Personalprono...

1.3 Übungen: Einfache Dialog...

1.4 Abschlusstest

Kapitel 2: Nomen und Artikel


2.1 Wortschatz: Häufige Nomen

2.2 Grammatik: Geschlecht un...

2.3 Übungen: Lückentexte

2.4 Abschlusstest

Kapitel 3: Grundlegende Verben



Ein Bereich ähnelt einer Ordnerstruktur und ermöglicht es Ihnen, eine Reihe von Elementen und Komponenten in einem Menü zu gruppieren.

Neuen Bereich Hinzufügen

## A-3 Deluge Funktionen Teil 1

HTML-Snippet

HTML Snippet

X

Deluge

Versions

Referenzfelder

Embed

Form

Report

Widget

Condition

if

else if

else

conditional if

Data Access

fetch records

aggregate records

update record

update multiple fields

for each record

Miscellaneous

set variable

call function

add comment

info

```
1 <{%  
2 %>  
3 <style>  
4 .table-of-contents {  
5   border: 1px solid #0ea5e9;  
6   border-radius: 5px;  
7   padding: 20px;  
8   max-width: 600px;  
9   margin: 20px auto 80px;  
10  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);  
11 }  
12  
13 #table-of-contents-header {  
14   margin-bottom: 40px;  
15 }  
16  
17 .table-of-contents a {  
18   display: block;  
19   color: black;  
20   margin-bottom: 20px;  
21 }  
22  
23 .table-of-contents a:hover {  
24   color: #0ea5e9;  
25 }  
26  
27 .txt-align-left {  
28   text-align: left;  
29 }  
30  
31 .bullet-free {  
32   list-style-type: none;  
33 }  
34  
35 .final-test li input {  
36   border: 1px solid #495057;  
37   border-radius: 5px;  
38 }
```

## A-4 Deluge Funktionen Teil 2

HTML-Snippet  
HTML Snippet

Deluge

VersionsReferenzfelder

List Manipulation

create list

add

remove index

remove element

add all | remove all

clear list

sort

for each element

for each index

Map Manipulation

create map

put

put all keys

remove key

clear map

Web Data

zoho integration

other integration

get url

```
1 <%{
2   %>
3 <style>
4 .table-of-contents {
5   border: 1px solid #0ea5e9;
6   border-radius: 5px;
7   padding: 20px;
8   max-width: 600px;
9   margin: 20px auto 80px;
10  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
11 }
12
13 #table-of-contents-header {
14   margin-bottom: 40px;
15 }
16
17 .table-of-contents a {
18   display: block;
19   color: black;
20   margin-bottom: 20px;
21 }
22
23 .table-of-contents a:hover {
24   color: #0ea5e9;
25 }
26
27 .txt-align-left {
28   text-align: left;
29 }
30
31 .bullet-free {
32   list-style-type: none;
33 }
34
35 .final-test li input {
36   border: 1px solid #495057;
37   border-radius: 5px;
38 }
```

## A-5 Deluge Funktionen Teil 3

post url	27	.txt-align-left {
	28	text-align: left;
	29	}
invoke url	30	
	31	.bullet-free {
open url	32	list-style-type: none;
	33	}
invoke API	34	
	35	.final-test li input {
Xml Manipulation	36	border: 1px solid #495057;
	37	border-radius: 5px;
execute XPath	38	}

## A-6 HC-Kontaktformular

---

A1

A2

B1

B2

C1

**Kontakt**

Chat

Schreiben Sie uns

Vorname

Nachname

E-Mail

Nachricht

Abandonen

## A-7 HC-Kontaktformular Code

```
<form>
  <div>
    <h4 class="text-center mb-5">Schreiben Sie uns</h4>
    <div class="p-2 contact-row">
      <label for="vorname">Vorname</label>
      <input type="text" id="vorname" class="contact-input" />
    </div>
    <div class="p-2 contact-row">
      <label for="nachname">Nachname</label>
      <input type="text" id="nachname" class="contact-input" />
    </div>
    <div class="p-2 contact-row">
      <label for="email">E-Mail</label>
      <input
        type="text"
        name="email"
        id="email"
        class="contact-input" />
    </div>
    <div class="p-2 contact-row">
      <textarea name="" id="" class="contact-input contact-msg">
        Nachricht</textarea>
    </div>
  </div>
</form>
```



## A-8 Feldeigenschaften von Formularfeldern

LiveCode  
Kontakt

Fertig

Name

E-Mail

Adresse

Telefon

Einzeilig

Mehrzeilig

123  
Zahl

Datum

Zeit

Auswahlliste

Radio

Mehrfachauswahl

Ihre Anmerkungsinhalte werden hier angezeigt

Vorname

Nachname

Email

20

Feldeigenschaften

Anmerkungen

<div style="text-align: center; font-size: 20px">Schreiben Sie uns </div></div>

Feldlinkname

plain

Feldreferenzen anzeigen

Aussehen

Feldtyp

Anmerkungen hinzufügen

72

## A-9 HTML-Editor im Form Builder

Edit HTML

```
<div style="text-align: center; font-size: 20px">  
  Schreiben Sie uns  
  <br>  
</div>
```

Schreiben Sie uns

Apply changes

Cancel

## A-10 HTML-NC-Kontaktformular über den Zoho Creator

Kontakt

Schreiben Sie uns

Vorname

Nachname

Email

Absenden

Rücksetzen

## A-11 Anpassungsoptionen für den Formular-Baustein

## A-12 Präfix-, Suffix-, Pflichtfeld- und PBD-Angabe

creator.zoho.eu/appbuilder/erahman/lexicode/formbuilder/Kontakt/edit

Get a sneak peek at the powerful new features and enhancements coming your way in March 2025! [Read Post](#)

Hide Close Fertig

LexiCode Kontakt

Basisfelder

- Name
- E-Mail
- Adresse
- Telefon
- Einzeilig
- Mehrzeilig
- Zahl
- Datum
- Zeit
- Auswahlliste
- Radio
- Mehrfachauswahl
- ☒

Ihre Anmerkungsinhalte werden hier angezeigt

Vorname

Nachname

Email

Feldeigenschaften

Feldname  
Vorname

Feldlinkname  
Vorname

Feldreferenzen anzeigen

Validierung

☐ Pflichtangabe

Felder anzeigen

☐ Prefix

☒

☐ Last Name

☐ Suffix

Datenschutz

☒ Enthält personenbezogene Daten (PD)

Datensicherheit

Die Datenverschlüsselung wird nicht unterstützt.  
Sichtbarkeit

### A-13 Feldtyp-, Feldgrößen- und Beschreibungstextangabe

Get a sneak peek at the powerful new features and enhancements coming your way in March 2025! [Read Post](#)

UnCode  
Kontakt

+

Basisfelder

 Name	 E-Mail
 Adresse	 Telefon
 Einzeilig	 Mehrzeilig
 Zahl	 Datum
 Zeit	 Auswahlliste
 Radio	 Mehrfachauswahl

Ihre Anmerkungsinhalte werden hier angezeigt

Vorname

Nachname

Email

Feldeigenschaften

Datenschutz

Enthält personenbezogene Daten (PDI)

Datensicherheit

Die Datenverschlüsselung wird nicht unterstützt.

Sichtbarkeit

Feld anzeigen für

Jeden

Aussehen

Feldtyp

Name

Bildgröße

Groß

Beschreibung

Anzeigen

Keine Beschreibung

Beschreibungstext

Beschreibung

Rest: 255

## A-14 HC-Lückentest

127.0.0.1:5500/A1/Interkapitel/Kapitel%201/a1-1.4-abschlusstest.html

[A1](#)  
[A2](#)  
[B1](#)  
[B2](#)  
[C1](#)  
[Kontakt](#)

### Abschlusstest A1 Wortschatz

1. Übersetze die folgenden Sätze ins Deutsche:

a) My name is Anna.

b) I am 25 years old.

c) Where are you from?

d) How old are you?

e) Goodbye and good night!

2. Fülle die Lücken mit den richtigen Wörtern aus:

a)  Morgen! (Good morning!)

b) Ich  30 Jahre alt. (I am 30 years old.)

c) Das  mein Freund Paul. (This is my friend Paul.)

d)  heißt du? (What is your name?)

e) Danke! -  (Thank you! - You're welcome!)

## A-15 LC-Lückentest im Zoho-Creator

The screenshot shows a web browser displaying the LexiCode application. The browser's address bar shows the URL: `creatorapp.zoho.eu/erahmiev/lexicode/#Page:Abschlusstest`. The application has a dark sidebar on the left with a menu containing items like 'A1', 'Index A1', '1.1 Wortschatz: Häufig...', '1.2 Grammatik', '1.3 Übungen: Einfach...', '1.4 Abschlusstest' (highlighted), 'A2', 'B1', 'B2', 'C1', 'Kontakt', and 'Prototyp'. The main content area is titled 'Abschlusstest A1 Wortschatz'. It contains two sections of exercises. Section 1, 'Übersetze die folgenden Sätze ins Deutsche:', has five items (a-e) with text input fields. Section 2, 'Fülle die Lücken mit den richtigen Wörtern aus:', has five items (a-e) with dropdown menus. A blue 'Abgeben' button is located between the two sections. At the bottom of the sidebar, there is a user profile icon and the name 'Ervin Rahmiev'. The browser's top bar shows standard navigation icons and a 'Hilfe' button.

**Abschlusstest A1 Wortschatz**

1. Übersetze die folgenden Sätze ins Deutsche:

a) My name is Anna.

b) I am 25 years old.

c) Where are you from?

d) How old are you?

e) Goodbye and good night!

**Abgeben**

2. Fülle die Lücken mit den richtigen Wörtern aus:

a)  Morgen! (Good morning!)

b) Ich  30 Jahre alt. (I am 30 years old.)

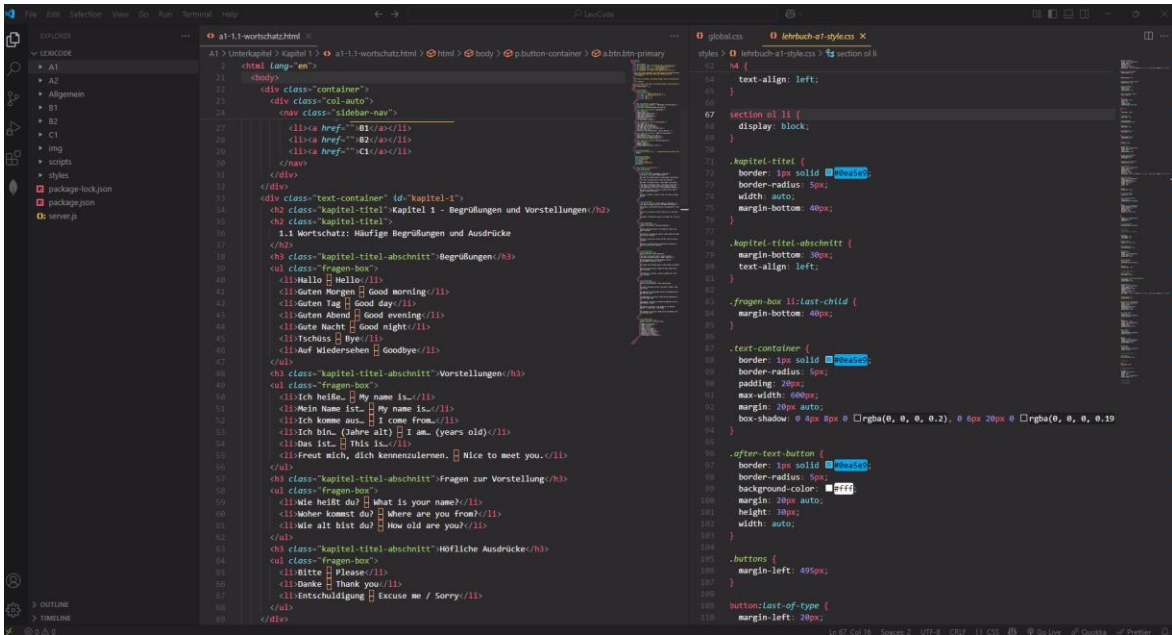
c) Das  mein Freund Paul. (This is my friend Paul.)

d)  heißt du? (What is your name?)

e) Danke! -  (Thank you! - You're welcome!)



## A-16 HTML- und CSS-Code als getrennte Dokumente



## A-17 HTML-Snippet mit HTML- und CSS-Code

```
131     background-color: #dddddd;
132 }
133
134 tr:hover {
135     color: #fff;
136     background-color: #0ea5e9;
137 }
138
139 .guide-box {
140     text-align: left;
141     line-height: 1.5;
142 }
143 </style>
144
145 <section>
146 <nav class="table-of-contents">
147 <h2 id="table-of-contents-header">Inhaltsverzeichnis</h2>
148 <ul class="content-box"> <h4>Kapitel 1: Begrüßungen und Vorstellungen</h4>
149 <li>1.1 Wortschatz: Häufige Begrüßungen und Ausdrücke</li>
150 <li>1.2 Grammatik: Personalpronomen</li>
151 <li>1.3 Übungen: Einfache Dialoge und Übungen</li>
152 </ul>
153 <ul class="content-box"> <h4>Kapitel 2: Nomen und Artikel</h4>
154 <li>2.1 Wortschatz: Häufige Nomen (Familie, Essen usw.)</li>
155 <li>2.2 Grammatik: Geschlecht und Pluralformen</li>
156 <li>2.3 Übungen: Lückentexte</li>
157 </ul>
158 <ul class="content-box"> <h4>Kapitel 3: Grundlegende Verben</h4>
159 <li>3.1 Wortschatz: Regelmäßige und unregelmäßige Verben</li>
160 <li>3.2 Grammatik: Konjugation im Präsens</li>
161 <li>3.3 Übungen: Konjugationsübungen und Satzbildung</li>
162 </ul>
163 <ul class="content-box"> <h4>Kapitel 4: Satzstruktur</h4>
164 <li>4.1 Wortschatz: Häufige Phrasen</li>
165 <li>4.2 Grammatik: Wortstellung und Fragesätze bilden</li>
166 <li>4.3 Übungen: Fragen basierend auf Vorgaben erstellen</li>
167 </ul>
168 <ul class="content-box"> <h4>Kapitel 5: Negation</h4>
```

## Abschlusstest A1 Wortschatz

1. Übersetze die folgenden Sätze ins Deutsche:

a) My name is Anna.

b) I am 25 years old.

c) Where are you from?

d) How old are you?

e) Goodbye and good night!

Abgeben

## Abschlusstest A1 Wortschatz

1. Übersetze die folgenden Sätze ins Deutsche:

a) My name is Anna.

b) I am 25 years old.

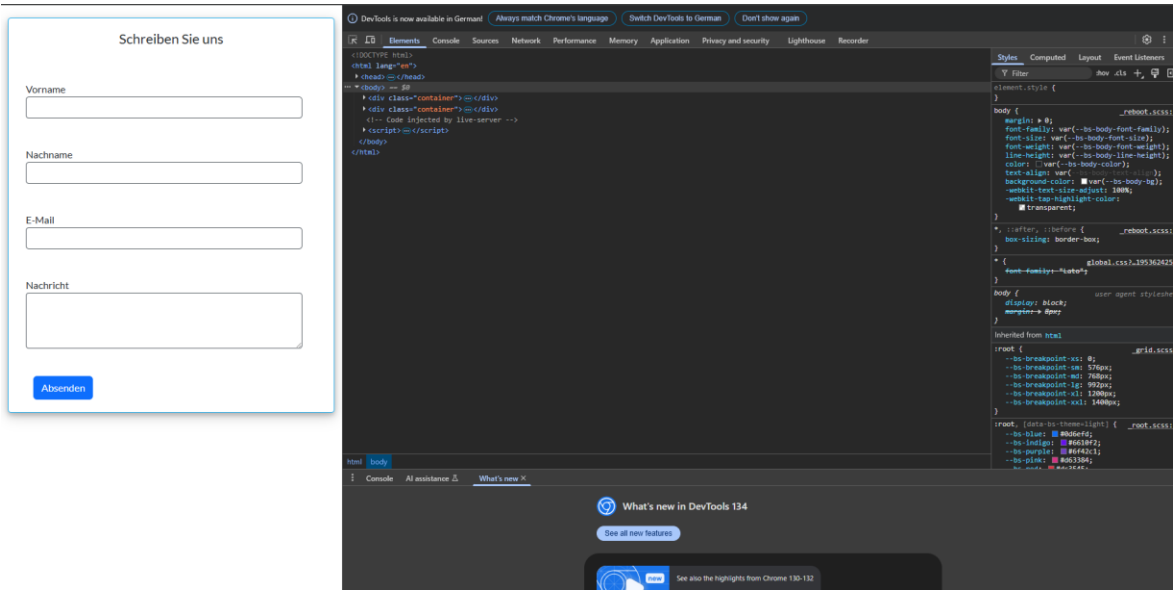
c) Where are you from?

d) How old are you?

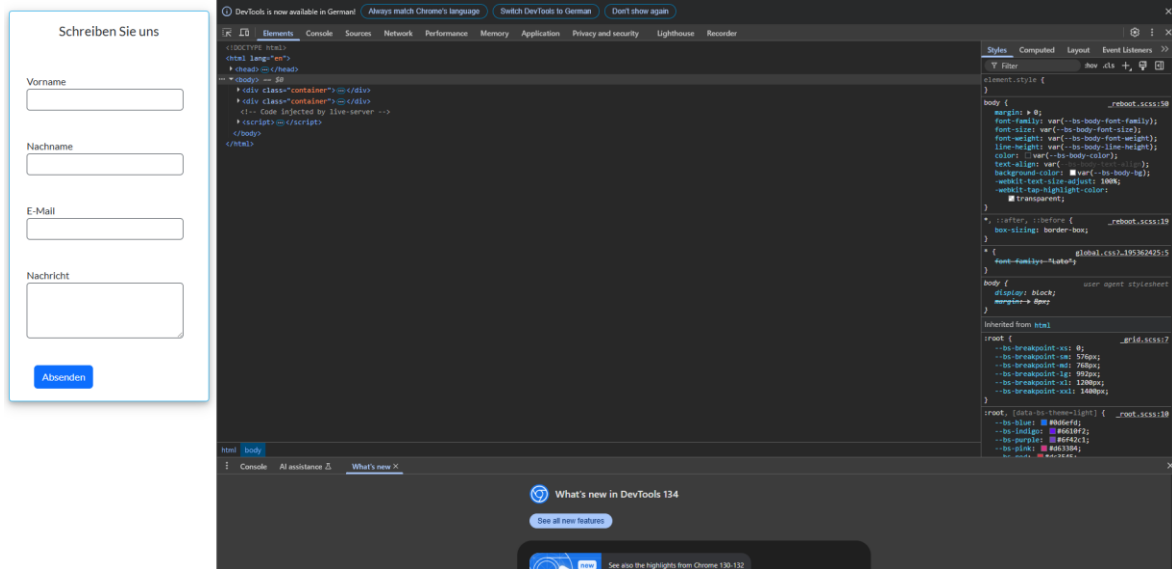
e) Goodbye and good night!

Abgeben

# A-20 HC-Kontaktformular bei 550 Pixel Bildschirmbreite



## A-21 HC-Kontaktformular bei 350 Pixel Bildschirmbreite



## A-22 LC-Abschlusstest bei 550 Pixel Bildschirmbreite

The screenshot shows a web browser window with the URL `creatorapp.zoho.eu/erahmiev/lexicode/#Page:Abschlusstest`. The application 'LexiCode' has a dark sidebar with a menu. The main content area displays the 'Abschlusstest A1 Wortschatz' (Final Test A1 Vocabulary). The test consists of five translation questions (a-e) with input fields. A blue 'Abgeben' (Submit) button is at the bottom. The right side of the browser shows the DevTools interface with the 'Styles' panel open, displaying CSS rules for the application's theme.

Get ready for incredible developments in 2025! Release Projection 1 is packed with exciting features that will transform your experience!

LexiCode

Abschlusstest A1 Wortschatz

1. Übersetze die folgenden Sätze ins Deutsche:

a) My name is Anna.

b) I am 25 years old.

c) Where are you from?

d) How old are you?

e) Goodbye and good night!

Abgeben

DevTools is now available in German!

Always match Chrome's language

Switch DevTools to German

Don't show again

Styles

Computed

Layout

Filter

element.style {

}.zc-micsbanner {

overflow: hidden;

}.zc-custom-font, .zc-custom-font

button, html, body, div, span,

textarea, applet, object, iframe,

h1, h2, h3, h4, h5, h6, p,

blockquote, pre, a, abbr, acronym,

address, big, cite, code, del,

dfn, em, img, ins, kbd, q, s,

samp, small, strike, strong, sub,

sup, tt, var, b, u, i, center, dl,

dt, dd, ol, ul, li, fieldset,

form, label, legend, table,

Console

What's new

What's new in DevTools 134

See all new features

## Appendix B. Tabellen

### B-1 Aufwandsermittlungswerte bei der HC-Version von Lexicode

Messgröße	Wert
Azahl der Solltestfälle (TF)	71,75
angestrebte Testüberdeckung (TU)	0,67
manuelle Testproduktivität (TP)	4
Grad der Testautomatisierung (TA)	0,4
Testbarkeitsmetrik (TB)	0,45
mittlere Testbarkeit (MT)	0,5
Testskalierungsexponent (TE)	1,03
Justierungsfaktor (AF)	1



## B-2 Ermittlung des wahrscheinlichen Schadens bei der HC-Version

Messgröße	Berechnung	Ergebnis
Maximaler Schaden	$0,861 \times (100 \text{ ha} \times 20 \text{ €}) = 1.722 \text{ €}$ $4,3 \times (50 \text{ ha} \times 20 \text{ €}) = 4.300 \text{ €}$ $1.722 \text{ €} + 4.300 \text{ €}$	6.022 €
Minimaler Schaden	$0,861 \times (50 \text{ ha} \times 20 \text{ €}) = 861 \text{ €}$ $4,3 \times (10 \text{ ha} \times 20 \text{ €}) = 860 \text{ €}$ $861 \text{ €} + 860 \text{ €}$	1.721 €
Wahrscheinlicher Schaden	$(6.022 \text{ €} + 1.721 \text{ €}) / 2$	3.871 €

### B-3 Aufwandsermittlungswerte bei der LC/NC-Version von Lexicode

Messgröße	Wert
Anzahl der Solltestfälle (TE)	74,35
angestrebte Testüberdeckung (TU)	0,67
manuelle Testproduktivität (TP)	4
Grad der Testautomatisierung (TA)	0,4
Testbarkeitsmetrik (TB)	0,45
mittlere Testbarkeit (MT)	0,5
Testskalierungsexponent (TE)	1,03
Justierungsfaktor (AF)	1

#### B-4 Ermittlung des wahrscheinlichen Schadens bei der LC/NC-Version

Messgröße	Berechnung	Ergebnis
Maximaler Schaden	$0,89 \times (100 \text{ ha} \times 20 \text{ €}) = 1.780 \text{ €}$ $4,45 \times (50 \text{ ha} \times 20 \text{ €}) = 4.450 \text{ €}$ $1.780 \text{ €} + 4.450 \text{ €}$	6.230 €
Minimaler Schaden	$0,89 \times (50 \text{ ha} \times 20 \text{ €}) = 890 \text{ €}$ $4,45 \times (10 \text{ ha} \times 20 \text{ €}) = 890 \text{ €}$ $890 \text{ €} + 890 \text{ €}$	1.780 €
Wahrscheinlicher Schaden	$(6.230 \text{ €} + 1.780 \text{ €}) / 2$	4.005 €

### **Erklärung der Echtheit**

Hiermit erkläre ich, dass ich diese Bachelor-/Masterarbeit selbständig und ohne fremde Hilfe angefertigt habe. Ich habe nur die angegebenen Quellen und Hilfsmittel benutzt und alle Quellen angegeben, denen ich Text und Inhalt entnommen habe. Diese Arbeit oder Teile davon wurden noch nie einem anderen Prüfungsausschuss vorgelegt. Ich erkläre mich mit einer Plagiatsprüfung meiner Arbeit durch einen Plagiatserkennungsdienst einverstanden.

---

Ort, Datum

---


Unterschrift

### Erklärung der Echtheit

Hiermit erkläre ich, dass ich diese Bachelor-/Masterarbeit selbständig und ohne fremde Hilfe angefertigt habe. Ich habe nur die angegebenen Quellen und Hilfsmittel benutzt und alle Quellen angegeben, denen ich Text und Inhalt entnommen habe. Diese Arbeit oder Teile davon wurden noch nie einem anderen Prüfungsausschuss vorgelegt. Ich erkläre mich mit einer Plagiatsprüfung meiner Arbeit durch einen Plagiatserkennungsdienst einverstanden.

Mannheim, 30.04.2025

Ort, Datum



Unterschrift