

textOsFtextTOsFliningLFliningTLFtextosflininglftabulartabproportionalprop
superiorSup superiorSup
fontspechyperref



EXPOSÉ ZUR (FIKTIVEN) ABSCHLUSSARBEIT

Vergleichende Analyse der Cross-Platform-Frameworks Tauri und Electron

**Evaluierung hinsichtlich Artefaktgröße,
Deployment-Mechanismen und
Ressourceneffizienz im industriellen Kontext**

vorgelegt von:

Luca Michael Schmidt

betreut von:

Ludwig Loth

3. Dezember 2025

Zusammenfassung

Die Digitalisierung von Lieferketten erfordert Softwarelösungen, die sich nahtlos in heterogene IT-Landschaften integrieren lassen. Bei der Entwicklung einer Desktop-Anwendung zur Erfassung und Übermittlung von Produktspezifikationen (GBI) stehen Unternehmen vor der Herausforderung, Software an Lieferanten mit unbekannter Hardwareausstattung und restriktiven IT-Richtlinien zu verteilen. Etablierte Frameworks wie Electron bündeln eine vollständige Browser-Laufzeitumgebung, was zu hohen Dateigrößen führt und die Verteilung via E-Mail sowie die Installation ohne Administratorrechte erschwert.

Dieses Exposé skizziert das Vorhaben einer vergleichenden Analyse des Frameworks Tauri v2 gegenüber Electron. Ziel ist die Evaluierung, ob Tauri durch die Nutzung systemseitiger Webviews und eines Rust-basierten Backends die Anforderungen an minimale Artefaktgrößen und ressourcenschonenden Betrieb besser erfüllen kann. Ein besonderer Fokus wird auf der Anpassbarkeit des Installationsprozesses mittels NSIS (Nullsoft Scriptable Install System) für Umgebungen ohne privilegierte Nutzerrechte liegen. Die geplante Arbeit soll durch quantitative Messungen und eine Nutzwertanalyse eine fundierte Entscheidungsgrundlage für den Einsatz moderner Cross-Platform-Technologien in restriktiven B2B-Umfeldern liefern.

Inhaltsverzeichnis

Abbildungsverzeichnis	2
1 Einleitung und Motivation	3
1.1 Problemstellung	3
1.2 Motivation	3
2 Stand der Forschung	4
2.1 Electron als etablierter Standard	4
2.2 Alternative Ansätze: Flutter, Qt und Tauri	4
2.3 Forschungslücke: Tauri v2 im Enterprise-Kontext	5
3 Zielsetzung und Forschungsfragen	5
3.1 Forschungsfragen	6
3.2 Erwartete Resultate	6
4 Methodik und geplantes Vorgehen	6
4.1 Vergleichende Implementierung (Quantitative Methode)	6
4.2 Nutzwertanalyse der Deployment-Fähigkeiten (Qualitative Methode)	7
4.3 Evaluationskriterien	7
5 Zeitplan und Ressourcen	8
A Anhang	9
A.1 Vorläufige Gliederung der Abschlussarbeit	9
B Verzeichnis der verwendeten Werkzeuge	11
Literaturverzeichnis	12

Abbildungsverzeichnis

1	Architekturvergleich Electron vs. Tauri	4
---	---	---

1 Einleitung und Motivation

1.1 Problemstellung

Im Rahmen der Zusammenarbeit mit externen Zulieferern ist eine präzise Übermittlung von Produktspezifikationen (PSPs) essenziell. Zur Standardisierung dieses Prozesses wird das Tool *GBI* entwickelt, welches Lieferanten ermöglicht, komplexe Formulare auszufüllen, technische Zeichnungen zu synchronisieren und diese in einem validierten JSON-Format gebündelt als ZIP-Archiv per E-Mail zu versenden.

Die technische Verteilung dieser Software unterliegt jedoch strengen Restriktionen:

1. **Heterogene Hardware:** Die Ausstattung der Lieferanten ist dem Unternehmen nicht im Detail bekannt. Es muss davon ausgegangen werden, dass teilweise ältere Systeme (Windows 10) mit begrenzten Arbeitsspeicherressourcen zum Einsatz kommen.
2. **Verteilung und Bandbreite:** Da die Software teilweise über E-Mail-Verteiler oder in Regionen mit limitierter Internetbandbreite bereitgestellt wird, ist die Dateigröße des Installers ein kritischer Faktor.
3. **Installationsberechtigungen:** Viele Lieferanten verfügen auf ihren Firmenrechnern über keine Administratorrechte. Der Installationsprozess muss daher ohne Erhöhung der Privilegien (User Mode) durchführbar sein und gleichzeitig unternehmensspezifische Anpassungen (Lizenztexte, Branding) unterstützen.

Klassische Ansätze wie Electron lösen das Cross-Platform-Problem durch das Bündeln einer Chromium-Instanz, was jedoch typischerweise zu Installer-Größen von über 80 MB und hohem Speicherverbrauch führt [1].

1.2 Motivation

Das Framework Tauri v2 verspricht durch die Trennung von Frontend (Web-Technologien) und Backend (Rust) sowie die Nutzung der im Betriebssystem vorhandenen Webview (WebView2 unter Windows [2]) eine signifikante Reduktion der Artefaktgröße und des Ressourcenverbrauchs. Zudem bietet Tauri v2 eine tiefe Integration des *Nullsoft Scriptable Install System* (NSIS). Dies könnte die Erstellung maßgeschneiderter Setup-Routinen ermöglichen, die ohne Admin-Rechte funktionieren – ein Feature, das mit alternativen Installern (z.B. WiX) nur komplex umsetzbar ist. Die Motivation dieser Arbeit liegt in der wissenschaftlichen Überprüfung, ob diese technologischen Vorteile in der Praxis Bestand haben und die strengen Anforderungen des Anwendungsfalls erfüllen.

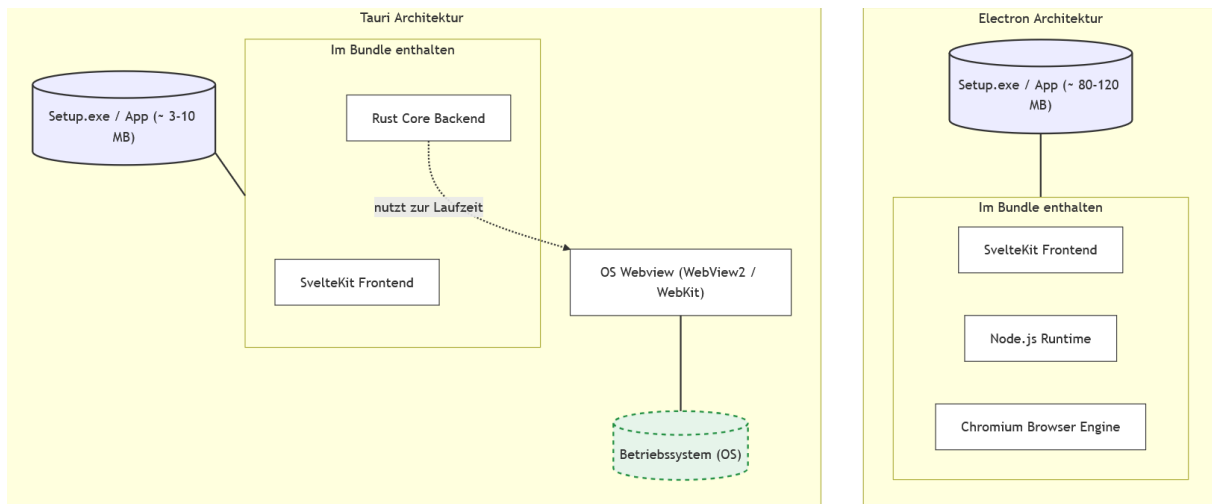


Abbildung 1: Architekturvergleich: Electron bündelt die Browser-Engine (Chromium), während Tauri auf die systemseitige Webview zugreift.

Quelle: Eigene Darstellung in Anlehnung an [3]

2 Stand der Forschung

Cross-Platform-Frameworks für Desktop-Anwendungen sind ein aktives Forschungsfeld. Die nachfolgende Darstellung skizziert den aktuellen Forschungsstand und ordnet die geplante Arbeit in den wissenschaftlichen Kontext ein.

2.1 Electron als etablierter Standard

Electron hat sich seit seiner Veröffentlichung 2013 als dominierendes Framework für plattformübergreifende Desktop-Anwendungen etabliert. Prominente Anwendungen wie Visual Studio Code, Slack oder Microsoft Teams basieren auf dieser Technologie [4]. Die Architektur kombiniert Chromium als Rendering-Engine mit Node.js für Backend-Funktionalität, wodurch Entwickler mit Web-Technologien vollwertige Desktop-Anwendungen erstellen können.

Thangadurai et al. [1] untersuchten in ihrer vergleichenden Studie den Energie- und Ressourcenverbrauch von Electron-basierten Kommunikationsanwendungen im Vergleich zu äquivalenten Web-Anwendungen. Die Autoren stellten fest, dass Electron-Anwendungen aufgrund der gebündelten Chromium-Engine typischerweise zwischen 80–120 MB Speicher im Leerlauf belegen und Installer-Größen von über 60 MB aufweisen. Dies stellt insbesondere in Umgebungen mit limitierter Bandbreite oder älteren Systemen eine Herausforderung dar.

2.2 Alternative Ansätze: Flutter, Qt und Tauri

Parallel zu Electron haben sich alternative Frameworks entwickelt, die unterschiedliche Strategien zur Reduzierung des Ressourcenverbrauchs verfolgen:

- **Flutter Desktop:** Googles UI-Framework nutzt die Dart-Sprache und rendert mit der eigenen Skia-Engine [5]. Dies ermöglicht native Performance, erfordert jedoch das Erlernen einer neuen Programmiersprache.
- **Qt:** Als langjähriges natives Framework bietet Qt plattformübergreifende Entwicklung in C++ [6]. Der Fokus liegt auf nativen Widgets, was zu kleineren Binaries führt, jedoch höhere Einstiegshürden mit sich bringt.
- **Tauri:** Das 2019 initiierte Framework verfolgt einen hybriden Ansatz, der Web-Technologien im Frontend mit Rust im Backend kombiniert [3]. Die zentrale Differenzierung zu Electron liegt in der Nutzung der betriebssystemeigenen Webview (WebView2 unter Windows, WebKit unter macOS/Linux) anstelle einer gebündelten Browser-Engine.

2.3 Forschungslücke: Tauri v2 im Enterprise-Kontext

Während die Architektur von Tauri in der technischen Dokumentation [3] beschrieben ist, existieren zum aktuellen Zeitpunkt nur wenige wissenschaftliche Studien, die Tauri – insbesondere die Version 2 mit erweiterten NSIS-Fähigkeiten – in einem industriellen B2B-Kontext evaluieren. Die Arbeit von Thangadurai et al. [1] liefert zwar Messmethodik für Performance-Vergleiche, bezieht Tauri jedoch nicht ein.

Die geplante Abschlussarbeit adressiert diese Lücke durch:

1. Eine quantitative Evaluierung der Ressourceneffizienz von Tauri v2 im Vergleich zu Electron anhand einer realen Anwendung.
2. Die Analyse der NSIS-Integration für Installationsszenarien ohne Administratorrechte – ein Aspekt, der in der bisherigen Literatur nicht behandelt wurde.
3. Die Überprüfung, ob die theoretischen Vorteile von Rust (Speichersicherheit, Performance) sich in praxisrelevanten Anwendungsfällen (ZIP-Komprimierung, Dateiverwaltung) messbar niederschlagen.

Damit leistet die Arbeit einen Beitrag zur empirischen Fundierung der Entscheidung zwischen modernen Cross-Platform-Frameworks in restriktiven Unternehmensumgebungen.

3 Zielsetzung und Forschungsfragen

Ziel der geplanten Abschlussarbeit ist die technische Evaluierung von Tauri v2 als Framework für industrielle Desktop-Anwendungen mit Fokus auf Ressourceneffizienz und Deployment-Flexibilität. Es soll geprüft werden, ob der Einsatz von Rust für dateiintensive Operationen

(ZIP-Erstellung, I/O) in Kombination mit einem SvelteKit-Frontend messbare Vorteile gegenüber einer äquivalenten Node.js-basierten Architektur (Electron) bietet [3].

3.1 Forschungsfragen

Zur Erreichung des Ziels werden folgende Forschungsfragen (FF) beantwortet:

- **FF1 (Artefaktgröße & Verteilbarkeit):** Wie verhalten sich die Dateigrößen der Installationsmedien von Tauri und Electron im Vergleich und wird die E-Mail-Versandfähigkeit durch Tauri gewährleistet?
- **FF2 (Ressourceneffizienz):** Welchen Einfluss hat die Auslagerung der Dateiverarbeitung (ZIP-Komprimierung, JSON-Generierung) in ein Rust-Backend auf die CPU-Last und den Arbeitsspeicherverbrauch im Vergleich zu einer Node.js-Laufzeitumgebung [4]?
- **FF3 (Installationsprozess):** Inwiefern ermöglicht die NSIS-Integration in Tauri v2 eine flexiblere Anpassung des Setups (Silent Install, Non-Admin Mode, Custom UI) im Vergleich zu Standard-Electron-Buildern?

3.2 Erwartete Resultate

Es wird erwartet, dass die Tauri-Anwendung eine Installer-Größe von unter 10 MB erreicht (vs. >60 MB bei Electron), was den E-Mail-Versand ermöglicht. Durch die Nutzung von Rust für die ZIP-Komprimierung wird eine performantere Abarbeitung erwartet als im Node.js Main-Process von Electron. Bezüglich des Installers soll gezeigt werden, dass NSIS die Anforderungen an eine benutzerfreundliche Installation ohne Administratorrechte vollständig erfüllt.

4 Methodik und geplantes Vorgehen

Die Arbeit folgt einem methodischen Mix aus **quantitativer Messung** und **qualitativer Analyse**. Dabei orientieren sich die Qualitätskriterien an der Norm ISO/IEC 25010 (Effizienz und Portabilität) [7].

4.1 Vergleichende Implementierung (Quantitative Methode)

Die Kernlogik der GBI-Anwendung ist bereits in Tauri (Rust/SvelteKit) implementiert. Für den wissenschaftlichen Vergleich wird ein **Referenz-Prototyp** in Electron erstellt. Dieser Prototyp bildet die kritischen Pfade der Anwendung nach:

1. Das Rendern der Formulare.
2. Das Generieren der JSON-Strukturen und das Packen des ZIP-Archivs (in Electron mittels Node.js-Modulen realisiert).

Anschließend werden auf einem Referenzsystem (Windows 10/11) standardisierte Messungen durchgeführt (Speicherverbrauch im Leerlauf, CPU-Peaks beim Zippen, finale Größe der .exe). Als Referenz für die Messmethodik dient die Studie von Thangadurai et al. [1].

4.2 Nutzwertanalyse der Deployment-Fähigkeiten (Qualitative Methode)

Es wird eine Nutzwertanalyse der Installer-Technologien durchgeführt. Die Bewertung erfolgt anhand folgender gewichteter Kriterien:

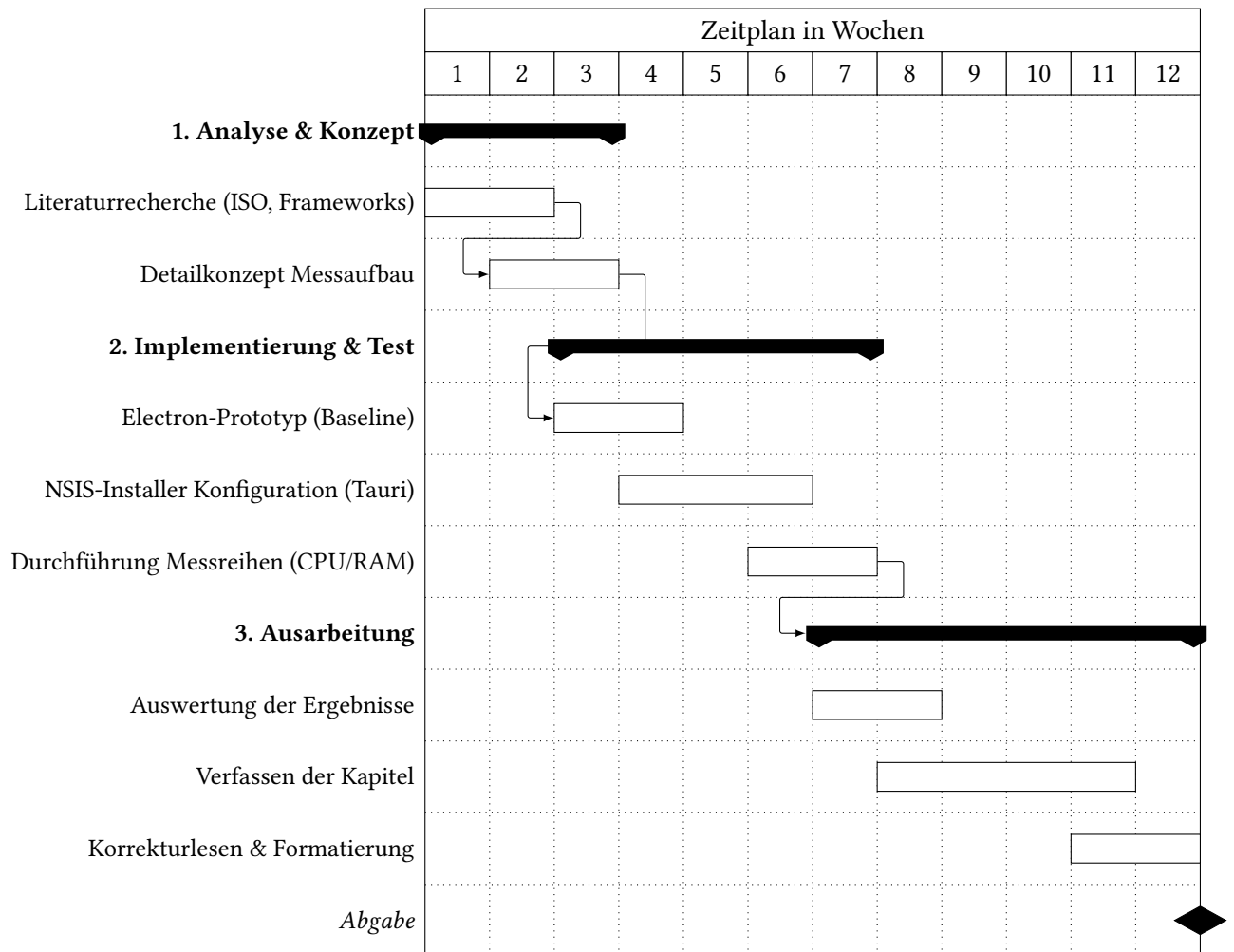
- **Konfigurationsaufwand** (Gewichtung 25%): Zeitaufwand und Komplexität für die initiale Konfiguration des Installers.
- **Anpassbarkeit** (Gewichtung 30%): Möglichkeit zur Integration von Lizenztexten, Branding und benutzerdefinierten Dialogen.
- **Installation ohne Admin-Rechte** (Gewichtung 35%): Funktionalität und Zuverlässigkeit der User-Mode-Installation.
- **Update-Mechanismus** (Gewichtung 10%): Unterstützung für automatische Updates und Code-Signierung.

Die Gewichtungsfaktoren reflektieren die Prioritäten des B2B-Anwendungsfalls, wobei die Installation ohne Administratorrechte als kritischste Anforderung eingestuft wird.

4.3 Evaluationskriterien

Die Bewertung erfolgt auf einer 5-stufigen Skala (1 = ungenügend, 5 = exzellent). Für jedes Framework wird der gewichtete Gesamtnutzen berechnet und dokumentiert. Zusätzlich wird der Update-Prozess (Signierung, Patch-Verteilung) evaluiert.

5 Zeitplan und Ressourcen



A Anhang

A.1 Vorläufige Gliederung der Abschlussarbeit

1. Einleitung

- 1.1 Motivation und Ausgangslage bei der Grenzebach BSH GmbH
- 1.2 Problemstellung: Herausforderungen bei der Softwareverteilung in heterogenen Lieferantennetzwerken
- 1.3 Zielsetzung der Arbeit
- 1.4 Forschungsfragen
- 1.5 Aufbau der Arbeit

2. Theoretische Grundlagen und Stand der Technik

- 2.1 Architekturmodelle für Cross-Platform-Desktop-Anwendungen
- 2.2 Analyse des Frameworks Electron (Node.js und Chromium)
- 2.3 Analyse des Frameworks Tauri v2 (Rust und System-Webview)
- 2.4 Technologien zur Software-Installation (NSIS vs. WiX vs. MSI)
- 2.5 Kriterien der Softwarequalität nach ISO 25010 (Fokus: Effizienz und Portabilität)

3. Konzeption der Vergleichsstudie

- 3.1 Definition des Anwendungsfalls: Das GBI-Tool
- 3.2 Anforderungsanalyse an den Rollout-Prozess (Silent Install, Non-Admin)
- 3.3 Definition der Messmetriken (Artefaktgröße, RAM, CPU, Startzeit)
- 3.4 Versuchsaufbau und Beschreibung der Testumgebung

4. Implementierung der Testumgebung

- 4.1 Entwicklung eines Referenz-Prototypen in Electron (Baseline)
- 4.2 Technische Umsetzung der GBI-Anwendung in Tauri
- 4.3 Konfiguration des NSIS-Installers für restriktive Benutzerrechte
- 4.4 Implementierung des Update-Mechanismus

5. Evaluation und Ergebnisse

- 5.1 Vergleich der Installer- und Anwendungsgrößen
- 5.2 Messergebnisse zur Laufzeitperformance (Speicher und CPU)

5.3 Qualitative Bewertung des Deployment-Prozesses und der Anpassbarkeit

5.4 Analyse der Kompatibilität auf verschiedenen Windows-Versionen

6. Diskussion

6.1 Interpretation der Messergebnisse im industriellen Kontext

6.2 Abwägung: Entwickler-Experience (Rust) vs. Performance-Gewinn

6.3 Risikobetrachtung: Abhängigkeit von der Webview2-Runtime

6.4 Handlungsempfehlung für die Grenzebach BSH GmbH

7. Fazit und Ausblick

7.1 Zusammenfassung der Ergebnisse

7.2 Ausblick: Portierung auf weitere Plattformen (macOS/Linux)

B Verzeichnis der verwendeten Werkzeuge

[**Gemini**] 3.0 Pro Preview, Google

Verwendung:

- Brainstorming zur Themenfindung und Abgrenzung (Präsentation vs. Exposé)
- Formulierungshilfen in: Kap. 1, Abs. 1.2, Kap. 3 und Kap. 4
- Generierung von Code für: Mermaid-Architekturdiagramm (Abb. 1)
- Literaturrecherche

[**Claude**] Sonnet 4.5, Anthropic

Verwendung:

- Formulierungshilfen in Kapitel 2
- Umstrukturierung der Methodik in Kap. 4
- Überprüfung der wissenschaftlichen Sprache und Stilistik
- Literaturrecherche

Literaturverzeichnis

- [1] J. Thangadurai, P. Saha u. a., “Electron vs. Web: A Comparative Analysis of Energy and Performance in Communication Apps,” in *Quality of Information and Communications Technology*, Springer, 2024, S. 177–193.
- [2] Microsoft Corporation, *Introduction to Microsoft Edge WebView2: Evergreen Distribution Mode*, Entwicklerdokumentation, 2025. besucht am 2. Dez. 2025. Adresse: <https://learn.microsoft.com/en-us/microsoft-edge/webview2/concepts/distribution>
- [3] Tauri Programme, *Tauri 2.0 Documentation: Distribution and NSIS Configuration*, Technische Dokumentation, The Tauri Programme, 2025. besucht am 2. Dez. 2025. Adresse: <https://v2.tauri.app/concept/architecture/>
- [4] OpenJS Foundation. “Electron Documentation: Application Architecture,” besucht am 3. Dez. 2025. Adresse: <https://www.electronjs.org/docs/latest/>
- [5] Google LLC. “Flutter Architectural Overview.” Technische Dokumentation, besucht am 3. Dez. 2025. Adresse: <https://docs.flutter.dev/resources/architectural-overview>
- [6] The Qt Company. “Qt 6 Documentation: Core Features and C++ APIs,” besucht am 3. Dez. 2025. Adresse: <https://doc.qt.io/qt-6/>
- [7] International Organization for Standardization, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE)*, 2011.

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Ich erkläre weiterhin, dass die vorliegende Arbeit in gleicher oder ähnlicher Form noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Bad Hersfeld, den 3. Dezember 2025

Luca Michael Schmidt