HOCHSCHULE FULDA
UNIVERSITY OF APPLIED SCIENCES

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 für heterogene IT-Landschaften. Bei der Entwicklung einer Desktop-Anwendung zur Erfassung und Übermittlung von Produktspezifikationen (GBI) müssen Unternehmen Software an Lieferanten mit unbekannter Hardwareausstattung und restriktiven IT-Richtlinien verteilen. Frameworks wie Electron bündeln eine vollständige Browser-Engine, was zu Dateigrößen von über 60 MB führt und die E-Mail-Verteilung sowie Installation ohne Admin-Rechte erschwert.

Dieses Exposé skizziert eine vergleichende Analyse von Tauri v2 und Electron. Die Evaluierung prüft, ob Tauri durch systemseitige Webviews und ein Rust-Backend die Anforderungen an minimale Artefaktgrößen und ressourcenschonenden Betrieb besser erfüllt. Ein Schwerpunkt liegt auf der Anpassbarkeit des Installationsprozesses mittels NSIS für Umgebungen ohne Admin-Rechte. Die Arbeit liefert durch quantitative Messungen und Nutzwertanalyse eine fundierte Entscheidungsgrundlage für moderne Cross-Platform-Technologien in B2B-Umgebungen.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung und Motivation

1.1 Problemstellung

Bei der Zusammenarbeit mit externen Zulieferern ist eine präzise Übermittlung von Produktspezifikationen (PSPs) essenziell. Das Tool *GBI* standardisiert diesen Prozess: Lieferanten können Formulare ausfüllen, technische Zeichnungen synchronisieren und diese in validiertem JSON-Format als ZIP-Archiv per E-Mail versenden.

Die technische Verteilung dieser Software unterliegt strengen Restriktionen:

1. **Heterogene Hardware:** Die Ausstattung der Lieferanten ist unbekannt. Teilweise kommen ältere Systeme (Windows 10) mit begrenzten Ressourcen zum Einsatz.
2. **Verteilung und Bandbreite:** Bei Verteilung über E-Mail oder in Regionen mit limitierter Internetbandbreite ist die Dateigröße kritisch.
3. **Installationsberechtigungen:** Viele Lieferanten verfügen über keine Administratorrechte. Die Installation muss im "User Mode" durchführbar sein und unternehmensspezifische Anpassungen (Lizenztexte, Branding) unterstützen.

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

1.2 Motivation

Tauri v2 nutzt einen Architekturansatz, der Frontend (Web-Technologien) und Backend (Rust) trennt sowie die systemeigene Webview (WebView2 unter Windows [2]) einsetzt. Dieser Ansatz lässt eine Reduktion von Artefaktgröße und Ressourcenverbrauch erwarten. Zudem bietet Tauri v2 eine tiefe Integration des *Nullsoft Scriptable Install System* (NSIS). Dies ermöglicht maßgeschneiderte Setup-Routinen ohne Admin-Rechte – ein Feature, das mit alternativen Installern (z.B. WiX) komplex umsetzbar ist. Diese Arbeit überprüft, ob diese technologischen Vorteile die strengen Anforderungen des Anwendungsfalls erfüllen.

2 Stand der Forschung

Cross-Platform-Frameworks für Desktop-Anwendungen sind ein aktives Forschungsfeld. Die folgende Darstellung ordnet die geplante Arbeit in den wissenschaftlichen Kontext ein.

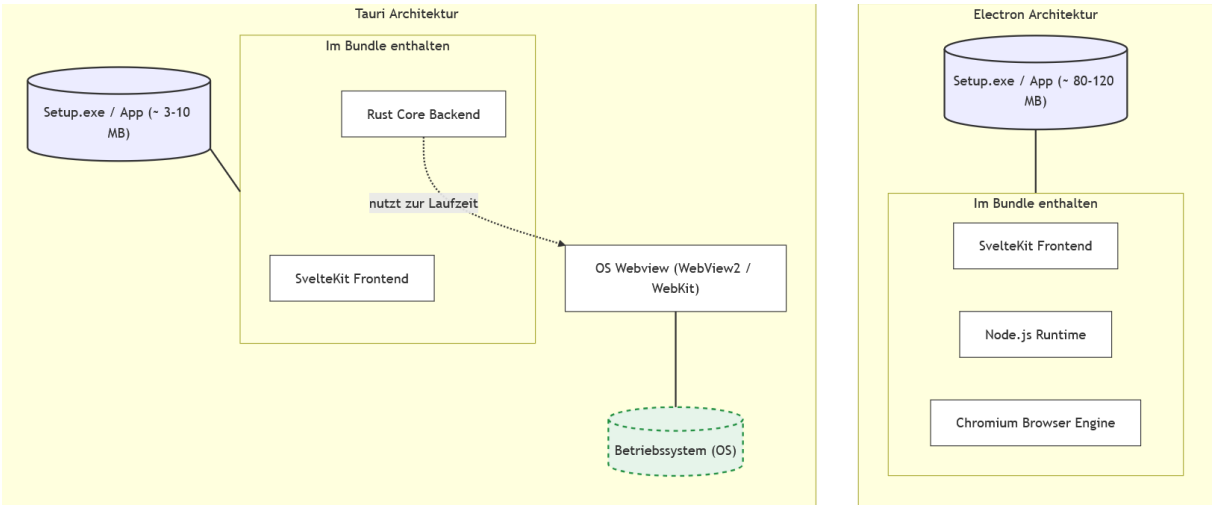

Abbildung 1: Architekturvergleich: Electron bündelt die Browser-Engine (Chromium), während Tauri auf die systemseitige Webview zugreift.

Quelle: Eigene Darstellung in Anlehnung an [3]

2.1 Electron als etablierter Standard

Electron hat sich seit 2013 als dominierendes Framework für plattformübergreifende Desktop-Anwendungen etabliert. Prominente Anwendungen wie Visual Studio Code, Slack oder Microsoft Teams nutzen diese Technologie [4]. Die Architektur kombiniert Chromium als Rendering-Engine mit Node.js für Backend-Funktionalität.

Thangadurai et al. [1] verglichen in ihrer Studie den Energie- und Ressourcenverbrauch von Electron-Anwendungen mit äquivalenten Web-Anwendungen. Sie stellten fest, dass Electron-Anwendungen aufgrund der gebündelten Chromium-Engine typischerweise 80–120 MB Speicher im Leerlauf belegen und Installer-Größen von über 60 MB aufweisen. Dies kann in Umgebungen mit limitierter Bandbreite oder älteren Systemen zu Einschränkungen führen.

2.2 Alternative Ansätze: Flutter, Qt und Tauri

Parallel zu Electron haben sich alternative Frameworks entwickelt, die unterschiedliche Strategien zur Reduzierung des Ressourcenverbrauchs verfolgen. Frameworks wie Flutter Desktop [5] und Qt [6] setzen auf native Rendering-Engines bzw. Widgets, erfordern jedoch die Beherrschung von Dart bzw. C++.

Tauri, seit 2019 entwickelt, verfolgt einen hybriden Ansatz: Web-Technologien im Frontend werden 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 kaum wissenschaftliche Studien, die Tauri v2 – insbesondere mit erweiterten NSIS-Fähigkeiten – in einem industriellen B2B-Kontext evaluieren. Die Arbeit von Thangadurai et al. [1] liefert Messmethodik für Performance-Vergleiche, bezieht Tauri jedoch nicht ein.

Die vorliegende Arbeit untersucht diese Lücke durch:

1. Quantitative Evaluierung der Ressourceneffizienz von Tauri v2 im Vergleich zu Electron anhand einer realen Anwendung.
2. Analyse der NSIS-Integration für Installationsszenarien ohne Administratorrechte.
3. Überprüfung, ob die Vorteile von Rust (Speichersicherheit, Performance) sich in praxisrelevanten Anwendungsfällen (ZIP-Komprimierung, Dateiverwaltung) messbar niederschlagen.

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

3 Zielsetzung und Forschungsfragen

Ziel der Abschlussarbeit ist die technische Evaluierung von Tauri v2 für industrielle Desktop-Anwendungen mit Fokus auf Ressourceneffizienz und Deployment-Flexibilität. Es wird geprüft, ob der Einsatz von Rust für dateiintensive Operationen (ZIP-Erstellung, I/O) mit SvelteKit-Frontend messbare Vorteile gegenüber Node.js-basierter 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?

- **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 30 MB erreicht (vs. >60 MB bei Electron), was E-Mail-Versand ermöglicht. Die Nutzung von Rust für ZIP-Komprimierung lässt eine performantere Abarbeitung im Vergleich zum Node.js Main-Process von Electron erwarten. NSIS soll eine benutzerfreundliche Installation ohne Admin-Rechte vollständig ermöglichen.

4 Methodik und geplantes Vorgehen

Die Arbeit folgt einem methodischen Mix aus **quantitativer Messung** und **qualitativer Analyse**. Die Qualitätskriterien orientieren sich an 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, der die kritischen Pfade abbildet:

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

Anschließend werden auf einem Referenzsystem (Windows 10/11) standardisierte Messungen durchgeführt (Speicherverbrauch im Leerlauf, CPU-Peaks beim Zippen, finale `.exe`-Größe). Messmethodik nach Thangadurai et al. [1].

4.2 Nutzwertanalyse der Deployment-Fähigkeiten (Qualitative Methode)

Die Installer-Technologien werden mittels Nutzwertanalyse bewertet. Gewichtete 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.

Die Bewertungsskala wird exemplarisch am Kriterium **Installation ohne Admin-Rechte** konkretisiert:

- **5 (exzellent):** Installation funktioniert zuverlässig ohne Admin-Rechte, vollständig dokumentiert, keine manuellen Anpassungen erforderlich.
- **4 (gut):** Installation ohne Admin-Rechte möglich, geringe manuelle Nacharbeiten oder Konfigurationsschritte nötig.
- **3 (befriedigend):** Installation möglich, jedoch mit Einschränkungen (z.B. eingeschränkte Funktionalität, Workarounds erforderlich).
- **2 (ausreichend):** Installation technisch möglich, aber mit erheblichem Aufwand oder instabil.
- **1 (ungenügend):** Installation ohne Admin-Rechte nicht möglich oder nicht praktikabel umsetzbar.

Für die übrigen Kriterien wird analog verfahren: Höhere Punktzahlen stehen für geringeren Aufwand (*Konfigurationsaufwand*), bessere Anpassungsmöglichkeiten (*Anpassbarkeit*) bzw. umfassendere Update-Funktionen (*Update-Mechanismus*).

5 Zeitplan und Ressourcen

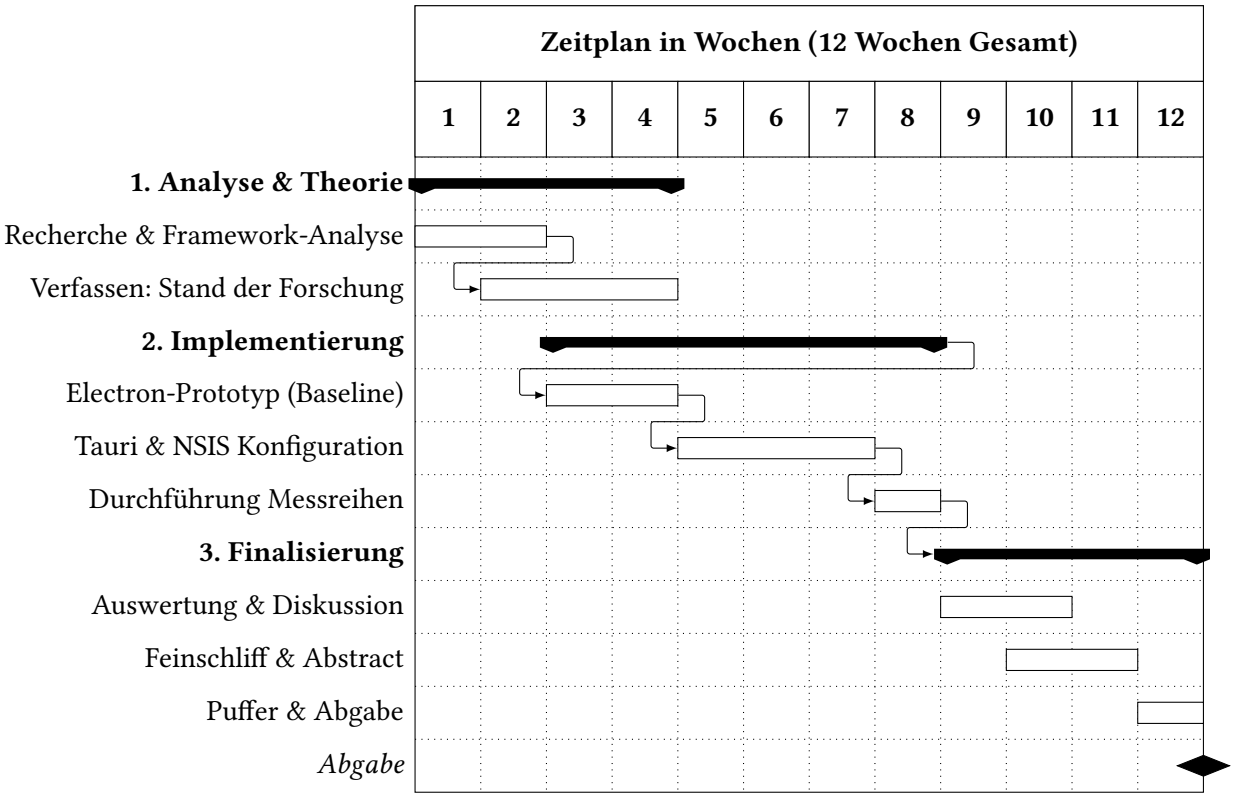

Abbildung 2: Zeitplan der Bachelorarbeit: Paralleles Verfassen der theoretischen Grundlagen während der Implementierungsphase.

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)

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
- Erstellung eines Basis-Entwurfs für: Mermaid-Architekturdiagramm (Abb. 1); anschließend manuell überarbeitet und gestylt.
  Prompt: `Erstelle ein Mermaid Graph Diagramm, das die Architektur von Electron (mit Node/Chromium) und Tauri (mit Rust/OS Webview) gegenüberstellt`
- Literaturrecherche

[Claude] Sonnet 4.5, Anthropic
Verwendung:

- Formulierungshilfen in Kapitel 2
- Umstrukturierung der Methodik in Kap. 4
- Überprüfung der wissenschaftlichen Sprache und Stilistik
- Reformulierung und Präzisierung in Kap. 1, 3, 4
- 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) — Product quality model*, 2024. besucht am 3. Dez. 2025. Adresse: https://www.iso.org/standard/78175.html

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