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

# Sperrvermerk

Die vorliegende Arbeit beinhaltet interne vertrauliche Informationen der Firma „Grenzebach BSH GmbH“. Die Weitergabe des Inhalts der Arbeit und Daten im Gesamten oder in Teilen ist grundsätzlich untersagt. Es dürfen keinerlei Kopien oder Abschriften - auch in digitaler Form - gefertigt werden. Ausnahmen bedürfen der schriftlichen Genehmigung der Firma Grenzebach BSH GmbH.

**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 erfüllt. Ein besonderer Fokus liegt auf der Anpassbarkeit des Installationsprozesses mittels NSIS (Nullsoft Scriptable Install System) für Umgebungen ohne privilegierte Nutzerrechte. Die Arbeit liefert damit eine Entscheidungsgrundlage für den Einsatz moderner Cross-Platform-Technologien in restriktiven B2B-Umfeldern.

# Inhaltsverzeichnis

**Abbildungsverzeichnis** **3**

**1 Einleitung und Motivation** **4**

1.1 Problemstellung . . . . . 4

1.2 Motivation . . . . . 4

**2 Zielsetzung und Forschungsfragen** **5**

2.1 Forschungsfragen . . . . . 5

**3 Methodik und geplantes Vorgehen** **6**

3.1 Vergleichende Implementierung (Quantitative Methode) . . . . . 6

3.2 Analyse der Deployment-Fähigkeiten (Qualitative Methode) . . . . . 6

**4 Erwartete Resultate** **7**

**5 Zeitplan und Ressourcen** **7**

**A Anhang** **8**

A.1 Vorläufige Gliederung der Abschlussarbeit . . . . . 8

**B Verzeichnis der verwendeten Werkzeuge** **10**

**Literaturverzeichnis** **11**

# Abbildungsverzeichnis

1 Architekturvergleich: Electron bündelt die Browser-Engine (Chromium), während Tauri auf die systemseitige Webview zugreift. . . . . 5

# 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 (Üser 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 **thangaduraiElectronVsWeb2024**.

## 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) 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.

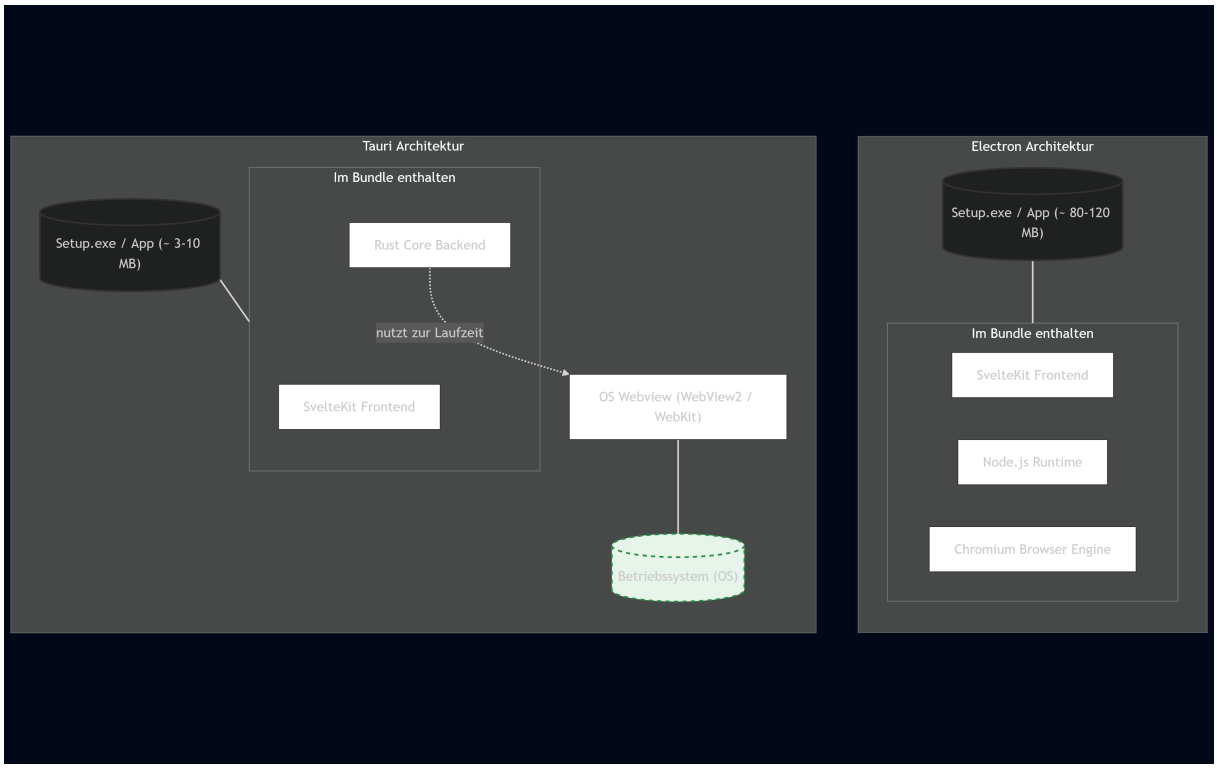



**Abbildung 1:** Architekturvergleich: Electron bündelt die Browser-Engine (Chromium), während Tauri auf die systemseitige Webview zugreift.

Quelle: Eigene Darstellung in Anlehnung an [tauriDocs2025]

# 2 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 **tauriDocs2025**.

## 2.1 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 **electronDocs**?

- **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 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 25010 (Effizienz und Portabilität) **iso25010**.

## 3.1 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. **thangaduraiElectronVsWeb2024**.

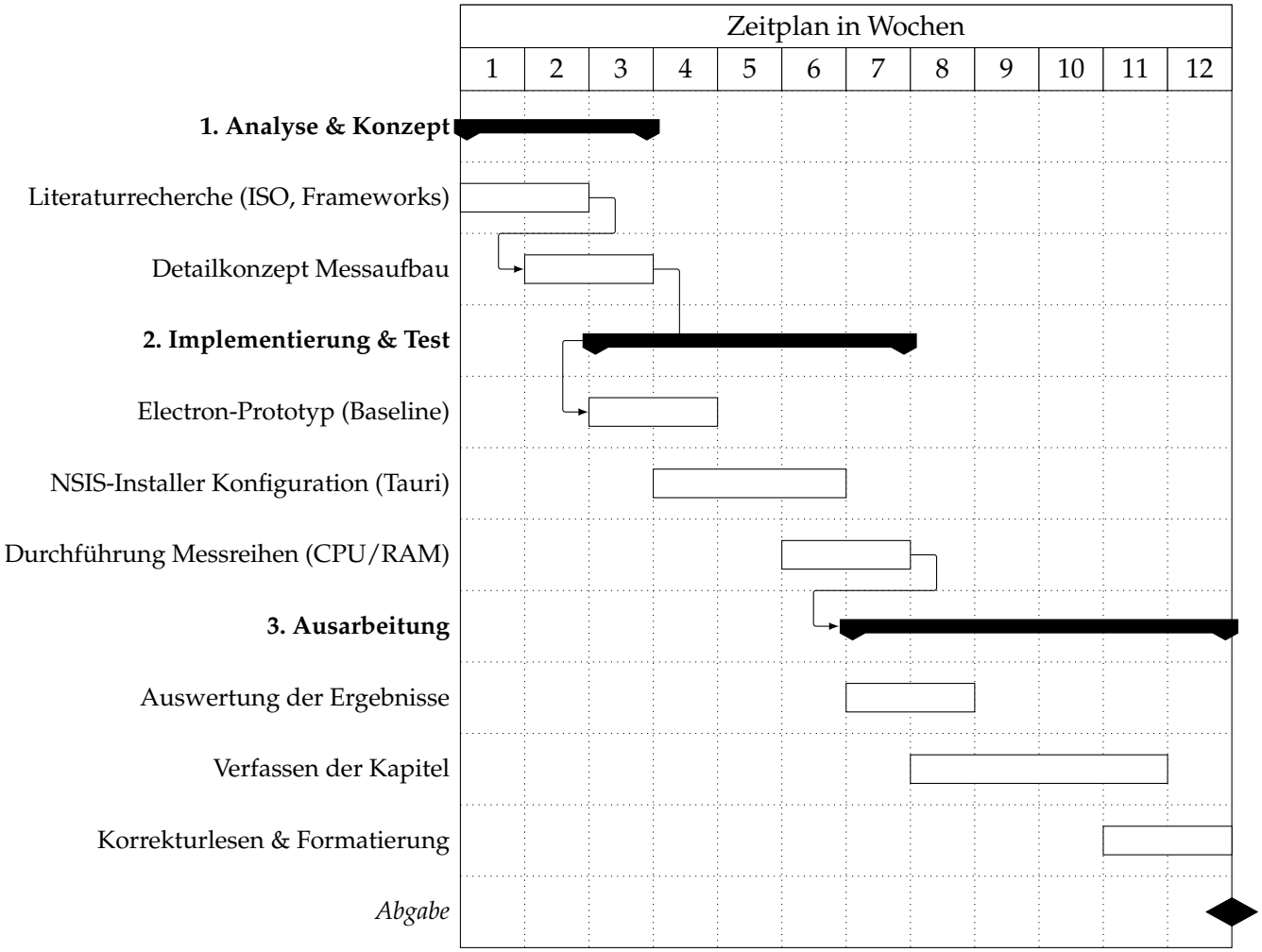
## 3.2 Analyse der Deployment-Fähigkeiten (Qualitative Methode)

Es wird eine Nutzwertanalyse der Installer-Technologien durchgeführt. Dabei wird untersucht, wie granular sich der NSIS-Installer in Tauri konfigurieren lässt (z.B. Einbinden von Lizenzvereinbarungen, Bildmaterial, Installationspfad-Wahl ohne Admin-Rechte) und wie sich dies zum Konfigurationsaufwand in Electron verhält. Zusätzlich wird der Update-Prozess (Signierung, Patch-Verteilung) evaluiert.

# 4 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.

# 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

Gemäß der KI-Richtlinie der Hochschule Fulda erkläre ich hiermit, dass zur Erstellung dieses Exposés folgende Systeme künstlicher Intelligenz (KI) unterstützend eingesetzt wurden. Die Verantwortung für den Inhalt liegt vollumfänglich beim Autor.

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

- Brainstorming zur Themenfindung und Abgrenzung (Präsentation vs. Exposé)
- Formulierungshilfen in: Kap. 1 (Einleitung), Abs. 1.2 (Motivation), Kap. 2 (Zielsetzung) und Kap. 3 (Methodik)
- Generierung von Code für: Mermaid-Architekturdiagramm (Abb. 1)
- Literaturrecherche

# 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