1. ASP Leaflet: Multi-Format Data Converter

Student: Arda Karadavut | **Projekt:** asp_cli | **Zeitraum:** Oktober 2025 - Januar 2026

1.1. Projektübersicht

Rust-Library zur bidirektionalen Konvertierung zwischen JSON, YAML, TOML und CSV. Bereitstellung als CLI-Tool und WebAssembly-Browser-App. Kernkonzept: Intermediate Representation reduziert 16 Konvertierungsfunktionen auf 8.

1.2. Lernerfolge

1.2.1. 1. Systemnahe Programmierung mit Rust

Lernprozess: Start am 10. Oktober 2025 mit Rust Book, ab November praktische Implementierung (minigrep-Projekt für File I/O und Error Handling). Iteratives Arbeiten: Implementieren, Testen, Refactoren mit gezielter KI-Unterstützung.

Erworbene Kenntnisse:

- **Ownership & Borrowing:** Speichersicherheit zur Compile-Zeit ohne Garbage Collector
- **Type-Safe Architecture:** FileFormat-Enum statt String-Matching – Compiler prüft alle 16 Kombinationen zur Compile-Zeit
- **Error Propagation:** Konsistentes Error-Handling mit `From` Trait und `?` Operator

Praktische Anwendung: Refactoring am 4. Januar 2026 zu Type-Safe Enums eliminierte ganze Klasse potentieller Runtime-Fehler.

Lernerfolg: Rust's Typ-System nutzen, um Fehler von Runtime zur Compile-Zeit zu verschieben.

1.2.2. 2. Architektur-Muster und Design-Entscheidungen

IR-Pattern: `serde_json::Value` als zentrale Zwischenrepräsentation. Jedes Format → IR → Zielformat. Reduziert Funktionen von 16 auf 8, macht System erweiterbar (neue Formate benötigen nur 2 statt 8 Funktionen).

Herausforderungen:

- **TOML-Limitation:** Root-Arrays nicht erlaubt → Automatisches Wrapping in Objekte
- **CSV-Hierarchie:** CSV flach, JSON/YAML/TOML hierarchisch → Flattening mit Unterstrichen

Lernerfolg: Erkennen, wann Abstractions Komplexität reduzieren. Trade-offs dokumentieren statt perfekte Lösung anzustreben.

1.2.3. 3. Agile Anpassungsfähigkeit

XML-Entscheidung: Nach Marktanalyse weggelassen – JSON, YAML, TOML, CSV decken 95% der Use Cases. Zeit in Web-Version investiert.

Ratzilla-Framework: Am 18. Dezember evaluiert, nach Neujahr als unreif erkannt. Am 3. Januar schnell auf HTML/CSS/JS UI umgeschwenkt.

Lernerfolg: Sunk-Cost-Fallacy vermeiden. Schnell auf alternative Lösungsansätze umschwenken statt an ursprünglichem Plan festzuhalten.

1.2.4. 4. WebAssembly und Cross-Platform-Deployment

Rust-Code wird durch `wasm-bindgen` zu WebAssembly kompiliert. `web-sys` ermöglicht DOM-Manipulation direkt aus Rust. Cargo-Features kompilieren denselben Core-Code für native CLI und WASM-Web – ein Bugfix funktioniert automatisch in beiden Versionen.

Lernerfolg: Libraries so designen, dass sie in verschiedenen Kontexten wiederverwendbar sind.

1.3. Erworbene Fachkompetenz

1.3.1. 1. Systemnahe Programmierung

Kompetenz: Entwicklung performanter, speichersicherer Anwendungen in Rust ohne Garbage Collector.

Nachweis: Ownership-System angewendet, Zero-Cost Abstractions genutzt, WASM-Kompilierung.

Relevanz: Backend-Services, DevOps-Tools, Performance-kritische Microservices.

1.3.2. 2. Datenformat-Transformation

Kompetenz: Analyse und Implementierung von Konvertierungs-Strategien für heterogene Systeme.

Nachweis: 16 bidirektionale Konvertierungen, Lösung komplexer Format-Limitationen, erweiterbare Architektur.

Relevanz: System-Integration, Daten-Migration, API-Gateways, ETL-Pipelines.

1.3.3. 3. Agiles Projektmanagement

Kompetenz: Mehrwert-orientierte Entscheidungsfindung, flexible Anpassung an technische Realitäten.

Nachweis: XML weggelassen, Ratzilla verworfen, Web-Version priorisiert – jede Entscheidung dokumentiert.

Relevanz: Iteratives Vorgehen bei unklaren Requirements, regelmäßige Mehrwert-Evaluation.

1.4. Quellen

Klabnik, S. and Nichols, C., 2025. *The Rust Programming Language*. (online) doc.rust-lang.org. Available at: link("https://doc.rust-lang.org/book/stable/") (Accessed 15 January 2026).

Tryzelaar, E. and Tolnay, D., 2014. *serde - Rust*. (online) docs.rs. Available at: link("https://docs.rs/serde/latest/serde/") (Accessed 15 January 2026).

The Rust CLI Working Group, 2025. *Command Line Applications in Rust*. (online) rust-cli.github.io. Available at: link("https://rust-cli.github.io/book/") (Accessed 15 January 2026).

Kontakt: Arda Karadavut | GitHub: link("https://github.com/Arda450")

ASP – Bachelor of Science (Hons.) Web Development – SAE Institute Zürich