Yaroslav Volokhodko (YVO 1070)



# Simple Salesman

## Projektdokumentation

Anwendung zur Adress- und Notizverwaltung im D2D-Vertrieb

Version 0.0.9

Stand: 02 Juli 2025

Autor: Yaroslav Vol.

GitHub: <https://github.com/Yaro-bit/SimpleSalesman>

## Executive Summary

Ziel:

Effiziente Verwaltung und Dokumentation von Door-to-door-Verkaufsaktivitäten durch eine leichtgewichtige, maßgeschneiderte Anwendung.

## Projektziele und Nutzen

### Hauptziele:

* Datenimport: Automatisierter wöchentlicher Import von Adress- und Auftragsdaten aus Excel-Dateien in eine PostgreSQL-Datenbank
* Datenerhalt: Bestehende Datensätze bleiben erhalten und sind pro Adresse mit Notizen erweiterbar
* DSGVO-konform: Vollständig lokal verarbeitet, mit verschlüsselter Speicherung und Zugriffsschutz
* Kleines Team, großer Effekt: Optimiert für ein kleines Vertriebsteam (2 Verkäufer) und einen Entwickler

### Kern-Nutzen:

* Zeitersparnis: Vermeidung manueller Eingabe – Reduktion der Dokumentationszeit um bis zu 75%
* Produktivitätssteigerung: Spart dem Vertriebsteam (2 Personen) ca. 4 Stunden pro Woche
* Kostenersparnis: Bei 25 €/Stunde sind das jährlich über 5.000 €
* Datenqualität: Einheitliche, strukturierte Datenbasis für Auswertungen und gezielte Folgeaktionen
* Transparenz & Kontrolle: Direkter Überblick über Besuchs Status, Notizen, Aufträge
* KI-Einsatz (optional): Strukturierte Textvorschläge durch ein kleines, lokal eingebettetes Modell

## Repräsentativer Entwickler Wert

Diese Software zeigt praxisnahe Kompetenz in:

* Full-Stack-Entwicklung (Spring Boot, PostgreSQL, REST-API, HTML/JS)
* Automatisierung von Geschäftsprozessen im Kleinbetrieb
* Umsetzung gesetzlicher Anforderungen (DSGVO, KI Act)
* Effiziente Architektur und Code-Struktur auch für kleine Teams

## Stakeholder Analyse

|  |  |  |
| --- | --- | --- |
| Stakeholder | Rolle | Hauptinteressen |
| Anwender - Verkäufer | Primärnutzer | Funktionale Anforderungen, Benutzerfreundlichkeit |
| Admin | Support | Support-Funktionen für den Verkäufer |
| DSGVO-Beauftragter | Compliance | Gesetzeslage und Verantwortlichkeit |
| KI Act Verantwortlicher | Compliance | Gesetzeslage (lokales Modell) |
| Lizenzverantwortlicher | Legal | Apache Lizenz Einhaltung |

## Relevante Referenzen und Compliance-Links

* RTR Regulierung: https://www.fb.gv.at/Info/rtr.html
* KI Act: https://artificialintelligenceact.eu/
* RTR Behörde: https://www.rtr.at/
* WKO KI-Richtlinien: https://www.wko.at/vlbg/information-consulting/kuenstliche-intelligenz
* WKO Datenschutz: https://www.wko.at/datenschutz/uebersicht

### Projektmanagement Aspekte

PM-Aufgaben:

* Marktanalyse und Benchmarking
* Vergleich mit bestehenden Lösungen der Konkurrenz
* Identifikation potenzieller Alleinstellungsmerkmale (z.B. DSGVO-/KI-Compliance, einfache Bedienung, Geschwindigkeit)

## Zielbestimmung

Das System "Simple Salesman" soll die manuelle Erfassung und Verwaltung von Door-to-Door-Vertriebsdaten durch einen automatisierten Import-Prozess und eine strukturierte, DSGVO-konforme Datenhaltung ersetzen. Die Lösung muss speziell auf die Bedürfnisse kleiner Vertriebsteams zugeschnitten sein und ohne umfangreiche Schulungen bedienbar sein.

## Produkteinsatz

|  |  |
| --- | --- |
| Aspekt | Beschreibung |
| Anwendungsbereich | Door-to-Door-Vertrieb in Österreich |
| Zielgruppen | Primär: 2 Vertriebsmitarbeiter im Außendienst  Sekundär: 1 Admin/Vertriebsleiter |
| Betriebsbedingungen | • Lokale Installation auf Windows/Linux Server• Zugriff über Webbrowser (Chrome, Firefox, Edge)• Tägliche Nutzung während Geschäftszeiten• Wöchentlicher Datenimport |

## Kern-Funktionen (Übersicht)

* Import von Excel-Dateien (wöchentlich)
* Speicherung und Abgleich der Daten in PostgreSQL
* Notizverwaltung pro Adresse
* Web-Frontend zur Auswahl, Anzeige und Bearbeitung
* Texteditor für die Felder
* Export von Excel-Dateien (Backup)
* Night Mode
* Modernes Design (ohne Frameworks)
* Wetter für die ausgewählte Region per REST
* Optional: Easter Eggs - Motivation Balken
* Optional: AI Text editor - Jira als Beispiel

## Funktionale Anforderungen

### USER - Anwender Funktionen

|  |  |  |
| --- | --- | --- |
| ID | Anforderung | Priorität |
| AF1 | Als Verkäufer möchte ich eine Excel-Datei hochladen können, um meine Adressdaten effizient zu importieren. | Muss |
| AF2 | Als Verkäufer möchte ich meine Adressen und Projekte anzeigen können, um meine Vertriebsaktivitäten zu überblicken. | Muss |
| AF3 | Als Verkäufer möchte ich zu jeder Adresse Notizen hinzufügen, bearbeiten und speichern können. | Muss |
| AF4 | Als Verkäufer möchte ich den Status eines Projekts aktualisieren können, um den Fortschritt nachvollziehen zu können. | Muss |
| AF5 | Als Verkäufer möchte ich Adressen und Projekte filtern und durchsuchen können, um gezielt Informationen zu finden. | Kann |
| AF6 | Als Verkäufer möchte ich meine Daten als Excel-Datei exportieren können, um ein Backup zu erstellen. | Kann |
| AF7 | Als Verkäufer möchte ich das aktuelle Wetter für eine bestimmte Region aufrufen können, um meine Routen besser zu planen. | Kann |
| AF8 | Als Verkäufer möchte ich optional KI-generierte Textvorschläge erhalten, um meine Notizen effizienter zu formulieren. | Wunsch |

### ADMIN - Admin Support Funktionen

|  |  |  |
| --- | --- | --- |
| ID | Anforderung | Priorität |
| AD1 | Als Admin möchte ich Benutzerkonten verwalten können, um den Zugang zur Anwendung zu kontrollieren. | Kann |
| AD2 | Als Admin möchte ich Fehlerprotokolle einsehen können, um Probleme schnell zu identifizieren. | Kann |
| AD3 | Als Admin möchte ich fehlerhafte Datenimporte rückgängig machen können, um Datenkorrektheit sicherzustellen. | Kann |
| AD4 | Als Admin möchte ich System-Backups und Wiederherstellungen durchführen können. | Kann |
| AD5 | Als Admin möchte ich eine Übersicht über alle Verkäufer und ihre Daten erhalten. | Kann |
| AD6 | Als Admin möchte ich gesetzlich relevante Informationen verwalten können, z.B. zur DSGVO. | Wunsch |

### DSGVO - Datenschutzanforderungen

|  |  |  |
| --- | --- | --- |
| ID | Anforderung | Priorität |
| DS1 | Die Anwendung muss alle Zugriffe und Änderungen an personenbezogenen Daten protokollieren. | Muss |
| DS2 | Alle personenbezogenen Daten müssen verschlüsselt gespeichert werden. | Muss |
| DS3 | Nutzer dürfen nur auf ihre eigenen Daten zugreifen können. | Muss |
| DS4 | Personenbezogene Daten müssen auf Anfrage vollständig gelöscht werden können (Recht auf Vergessenwerden). | Muss |
| DS5 | Die Anwendung muss eine Datenschutzerklärung anzeigen und die Einwilligung der Nutzer dokumentieren. | Muss |

### KI Act - Anforderungen an den KI-Einsatz

|  |  |  |
| --- | --- | --- |
| ID | Anforderung | Priorität |
| KI1 | KI-generierte Texte müssen als solche gekennzeichnet werden. | Muss |
| KI2 | Die Nutzung von KI-Funktionen muss abschaltbar sein (Opt-out). | Muss |
| KI3 | Die KI darf keine personenbezogenen Daten speichern oder weiterverarbeiten. | Muss |
| KI4 | Verwendete KI-Modelle müssen dokumentiert sein, inkl. Zweck und Versionsnummer. | Muss |

### Lizenz-Anforderungen - Apache License 2.0

|  |  |  |
| --- | --- | --- |
| ID | Anforderung | Priorität |
| LI1 | Ein Lizenzhinweis zur Apache License muss im Quellcode enthalten sein. | Muss |
| LI2 | Copyright-Hinweise externer Bibliotheken müssen übernommen werden. | Muss |
| LI3 | Alle verwendeten Open-Source-Komponenten müssen dokumentiert werden. | Muss |
| LI4 | Änderungen am Code müssen in einem Änderungsverlauf (CHANGELOG.md) nachvollziehbar dokumentiert werden. | Muss |

## Priorisierung der Anforderungen

Verpflichtende und optionale Anforderungen:

* Muss: AF1-AF4, DS1-DS5, KI1-KI4, LI1-LI4, NF1-NF5
* Kann: AF5, AF6, AF7, AD1-AD5
* Wunsch: AF8, AD6

## Technology Stack

|  |  |  |
| --- | --- | --- |
| Komponente | Technologie | Version/Details |
| Frontend | HTML, CSS, JavaScript | Vanilla JS, ohne Frameworks |
| Backend | Spring Boot | REST-API, Datenimport via Apache POI |
| Datenbank | PostgreSQL | Verschlüsselt - inklusive Hardware |
| Authentifizierung | Keycloak | OAuth 2.0 authorization |
| Build & Deployment | Maven, Git | CI/CD Pipeline |

## Systemarchitektur

Die Anwendung folgt einer klassischen 3-Schichten-Architektur mit optionaler Synchron- oder Asynchron-Verarbeitung:

* Frontend: UI zur Anzeige von Adressen & Notizen, Datei-Upload, API-Nutzung
* Backend: Datenimport, REST-API, Geschäftslogik, Kommunikation mit PostgreSQL
* Datenbank: Strukturierte Speicherung aller Daten mit normalisierten Relationen

## Datenmodell

### Entitäten und Beziehungen:

* Region → Adresse: 1:N
* Adresse → Projekt: 1:N
* Projekt → Status, Betreiber, Baufirma: jeweils 1:N

## Software Struktur

## Ablauf der Hauptprozesse

### Datenimport-Prozess:

1. Benutzer lädt Excel-Datei hoch
2. Backend verarbeitet und gleicht Daten mit PostgreSQL ab
3. Duplikate werden erkannt und zusammengeführt
4. Import-Log wird erstellt

### Datenbearbeitung:

1. Frontend zeigt aktuelle Daten interaktiv an
2. Notizen können ergänzt, bearbeitet oder gespeichert werden
3. Änderungen werden in Echtzeit synchronisiert
4. Audit-Trail protokolliert alle Änderungen

### Datenexport:

1. Gefilterte oder vollständige Daten werden exportiert
2. Excel-Format mit Zeitstempel
3. Backup-Funktionalität integriert

## Technische Optimierungen (DEV)

### Performance Optimierungen:

* JOIN / FETCH on database first 100 people: https://medium.com/javarevisited/spring-jpa-when-to-use-join-fetch-a6cec898c4c6
* 2 Parameter - Lazy/Eager Loading Strategien
* SQL-Analyzerzur Query-Optimierung
* Materialized Viewsfür häufige Abfragen (?)
* Native Queriesfür komplexe Operationen

### API & Dokumentation:

* OPEN APISpezifikation
* SWAGGER UIfür API-Testing
* Automatische API-Dokumentation

### Entwicklungstools:

* Cline (LLM open source)für Code-Assistenz
* Robocopyfür automatisierte Backups

## Weitere Entwicklungsaspekte

### Testing:

* Unit- und Integrationstests mit JUnit
* API-Tests mit REST Assured
* Frontend-Tests mit Selenium

### Dokumentation:

* API-Dokumentation (Swagger/OpenAPI)
* Benutzerdokumentation (Markdown)
* Technische Dokumentation (JavaDoc)
* Deployment-Handbuch

### Planung & Management:

* Git für Versionsverwaltung
* Gantt-Diagramme für Zeitplanung
* CI/CD Pipeline (Jenkins/GitLab CI)
* Regelmäßiges Refactoring

### Werkzeuge:

* draw.io/Lucidchart für ERM und UML
* GitHub für Versionierung und Issue-Tracking
* IntelliJ IDEA als Entwicklungsumgebung

### Deployment:

* Docker-Container für einfaches Deployment
* Umgebungsvariablen für Konfiguration
* Health-Checks und Monitoring
* Automatisierte Backup-Strategie

## Qualitätssicherung

### Code-Qualität:

* SonarQube Integration
* Code Reviews via Pull Requests
* Coding Standards (Java Code Conventions)
* Clean Code Prinzipien

### Sicherheit:

* OWASP Top 10 Compliance
* Regelmäßige Security Audits
* Penetration Testing
* Dependency Scanning

### Performance:

* Load Testing mit JMeter
* Response Time < 2 Sekunden
* Concurrent User Support (10+)
* Database Query Optimization

## 3.9 Offene Punkte und zukünftige Erweiterungen

### PM-Ebene:

* Detaillierte Marktanalyse ausstehend
* Benchmarking mit Konkurrenzprodukten
* Preismodell für kommerzielle Nutzung

### DEV-Ebene:

* Performance-Fixes nach Lasttests
* Deployment Configuration Manual finalisieren
* Feedback-Loop mit Anwendern etablieren
* UML-Diagramme (Use Case, Sequenz, Klassen) erstellen
* Evaluierung weiterer KI-Modelle für Textvorschläge

## Compliance und rechtliche Aspekte

Die Anwendung erfüllt alle relevanten gesetzlichen Anforderungen:

* DSGVO: Vollständige Compliance durch Verschlüsselung, Zugriffskontrolle und Löschfunktionen
* KI Act: Transparente KI-Nutzung mit Opt-out-Möglichkeit
* Apache License 2.0: Korrekte Lizenzierung und Attribution

## Software Stand 0.0.911

* Aktueller Stand: Version 0.0.9 (30.06.2025)
* Projekttyp: Funktionsfähige Webanwendung mit Frontend und Backend
* Status: Prototyp mit bekannten Performance-Problemen

## Funktionale Anforderungen (USER)

* AF1: Excel-Import vollständig implementiert (Apache POI, aber Performance-Probleme bei 70.000 Adressen)
* AF2: REST-API und Web-GUI (Thymeleaf, Bootstrap 5.3+), Performance-Einschränkungen vorhanden
* AF3: CRUD-Operationen für Notizen, API und Editor vollständig
* AF4: Statusänderung über API und GUI
* AF5: Teilweise implementiert, keine Filterung oder Pagination
* AF6: Teilweise implementiert, kein GUI-Zugriff
* AF7: Vollständig: Wetter-API, Geolokation, Fallback

## Technische Komponenten

* Frontend: HTML/CSS/JavaScript, Thymeleaf, Bootstrap 5.3.2, responsive, Icons, Modals, Spinners, Night Mode fehlt
* Backend: Spring Boot, REST, DTO-Validierung, Swagger, Exception Handling, Verschlüsselung
* Authentifizierung: Keycloak mit OAuth2 PKCE, JWT Token, automatisches Handling, Secure Logout
* Datenbank: PostgreSQL mit Verschlüsselung, vollständige JPA-Entities
* Testing & Dokumentation: Unit-Tests, Postman-Collection, englische Dokumentation, GDPR/AI Act vorhanden

# Teilweise implementierte Anforderungen

## DSGVO-Anforderungen

* DS1: Logging aktiviert (Spring Boot Standard)
* DS2: Verschlüsselung aktiv
* DS3: Zugriffskontrolle mit Keycloak
* DS4: Basis-Löschfunktion
* DS5: Noch keine UI für Consent-Management

# Noch nicht implementierte Anforderungen

## USER & ADMIN

* AF8: KI-generierte Vorschläge geplant (lokales GPT-2 Modell)
* AD2: Keine Fehlerprotokoll-GUI
* AD3: Kein Rollback für Imports
* AD4: Keine Backup/Restore GUI
* AD5: Keine Admin-Übersicht
* AD6: Keine DSGVO-Verwaltungsoberfläche

# Automatische Bewertung

## Positiv

* Funktionsfähige Anwendung mit Frontend & Backend
* Professionelle UI mit Bootstrap 5.3
* Aktive Authentifizierung mit Keycloak
* Responsive Design für mobile Nutzung
* Gute Dokumentation (GDPR/AI Act)
* Kern-Features vollständig

## Kritisch

* Performance-Probleme bei großen Datenmengen
* Keine Pagination, keine produktive Reife
* Fehlende DSGVO-GUI

## Fazit

Das Projekt hat sich von einem Backend-Prototyp (v0.0.5 – Stand davor) zu einer funktionsfähigen Webanwendung (v0.0.9) entwickelt. Die Kernfunktionalität ist implementiert, doch die Performance-Probleme verhindern produktiven Einsatz. Mit ca. 20-30 Stunden Entwicklungszeit könnten Pagination, Batching und Streaming das System produktionsreif machen.

## Zeiterfassung und Projektaufwand

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Stunden | Kumulativ | Phase |
| Davor | 10 | 10 | Konzeption |
| 28.05 | 4 | 14 |  |
| 29.05 | 6 | 20 |  |
| 31.05 | 3 | 23 |  |
| 09.06 | 5 | 28 |  |
| 15.06 | 4 | 32 |  |
| 16.06 | 4 | 36 |  |
| 17.06 | 6 | 42 |  |
| 18.06 | 7 | 49 | Entwicklung Backend |
| 20.06 | 2 | 51 |  |
| 21.06 | 3 | 54 |  |
| 26.06 | 6 | 60 | Entwicklung Frontend |
| 27.06 | 3 | 63 |  |
| 28.06 | 3 | 66 |  |
| 30.06 | 4 | 70 |  |
| 02.07 | 4 | 74 | Prototyp Vorbereitung |
| … | … | … | … |

## Repository Information

Repository: https://github.com/Yaro-bit/SimpleSalesman

Das Repository enthält:

* Vollständiger Quellcode
* Installationsanleitungen
* API-Dokumentation
* Beispieldaten für Tests
* CI/CD Konfiguration
* Issue Tracker für Bugs und Features

Hinweis: Diese Dokumentation stellt den Stand Juli 2025 dar.

Aktuelle Informationen und Updates finden sich im GitHub Repository.