# Projektübersicht – Softwareentwicklung & Projektmanagement (4. Jahrgang)

**Projektdauer:** 8 Wochen
**Unterricht:** 2 UE pro Woche
**Gruppengröße:** 3er- und 4er-Gruppen (Standard: 4er)

## 1. Projektidee (Rahmen)

Die Lehrperson stellt ein **vorgegebenes Git-Repository** bereit, das folgende Elemente enthält:

- Grundlegende Projektstruktur
- Basisklassen für Produkte/Objekte (Attribute, Grundmethoden)
- vorbereitete Module (Services, Ports/Adapter, UI-Skeleton)
- TOML-Setup (`pyproject.toml`) zur automatischen Erstellung der Entwicklungsumgebung (venv, Dependencies)

Auf dieser Basis entwickeln die Teams eine **verwaltende Software** (z. B. Lager-, Produkt- oder Objektverwaltung), die:

- grafisch bedienbar ist,
- Daten persistent verwaltet,
- systematisch getestet wird,
- im Team mittels Versionsverwaltung entwickelt wird.

Je Gruppe wird ein andere Thema vorgegeben, dass via eLearning eingeteilt wird. Folgende Themen stehen dabei zur Verfügung:

- Supermarkt
- Bäckerei
- Lagerverwaltung
- Autozuhändler
- Bürobedarf
- Fitnesscenter
- Baumarkt
- Buchhandel
- Paketverteilung

### 1.1 Zielsetzung des Projekts und Rolle der Programmierung

Ziel dieses Projekts ist nicht die möglichst umfangreiche Implementierung einer Software, sondern das Erlernen professioneller Arbeitsweisen in der Softwareentwicklung. Der Schwerpunkt liegt daher bewusst auf:

- strukturierter Zusammenarbeit im Team,
- sinnvoller Nutzung von **Versionsverwaltung (Git)**,
- Anwendung von **Dokumentations- und Projektmanagementstandards**,
- sauberer Definition und Einhaltung von **Schnittstellen (Contracts)**,
- sowie reflektiertem Einsatz von **KI-gestützten Werkzeugen**.

Die eigentliche Programmierleistung macht im Gesamtprojekt **nur etwa 30–40 %** des Arbeitsaufwands aus. Der restliche Anteil entfällt auf Planung, Dokumentation, Qualitätssicherung, Koordination, Versionierung und Reflexion.

Diese Gewichtung entspricht der Realität moderner Softwareprojekte: Auch in professionellen Entwicklungsteams besteht ein Großteil der Arbeit nicht aus reinem Schreiben von Code, sondern aus Abstimmung, Dokumentation, Wartbarkeit und Zusammenarbeit.

### 1.2 Einsatz von KI im Projekt

Der Einsatz von KI (z. B. zur Codegenerierung, Ideenfindung oder Fehleranalyse) ist **ausdrücklich erlaubt und erwünscht**, sofern folgende Grundsätze eingehalten werden:

- KI dient als **Unterstützungswerkzeug**, nicht als Ersatz für eigenes Verständnis.
- Ergebnisse müssen **verstanden, geprüft und ggf. angepasst** werden.
- Entscheidungen über Architektur, Schnittstellen und Projektstruktur liegen **immer beim Team**.
- Der Einsatz von KI ist **transparent zu dokumentieren**.

Gerade im Kontext leistungsfähiger KI-Systeme gewinnt die Fähigkeit, **Code einzuordnen, zu integrieren, zu testen, zu dokumentieren und weiterzuentwickeln**, stark an Bedeutung.
Dieses Projekt legt daher den Fokus auf genau jene Kompetenzen, die auch im KI-unterstützten Softwareengineering unverzichtbar sind.

## 2. Gruppengröße & Projektumfang

### 4er-Gruppen (Standardfall)

- **2 Reports (Pflicht)**
- GUI
- Tests (Unit- & Logiktests)
- vollständige PM-Dokumentation

### 3er-Gruppen

- **1 Report (Pflicht)**
- gleicher Qualitätsanspruch, reduzierter Umfang

## 3. Rollenverteilung (4er-Gruppe, verbindlich)

### Rolle 1 – Projektverantwortung & Schnittstellen (Contract Owner)

- Projektkoordination & Kommunikation
- **Zentrale Verantwortung für alle Schnittstellen (Contracts)**
- Dokumentation der Schnittstellen (`docs/contracts.md`)
- Integration der Komponenten
- Release- & Versionsverantwortung
- Unterstützung bei Mergekonflikten

### Rolle 2 – Businesslogik & Report A

- Implementierung der Kern-Use-Cases
- Umsetzung von **Report A** (z. B. Lagerstandsreport)
- zugehörige Tests

### Rolle 3 – Report B & Qualität (fällt in 3er Grp weg)

- Umsetzung von **Report B** (z. B. Bewegungsprotokoll, Statistik des Lagerverlaufs mit matplotlib)
- erweiterte Tests (Rand- & Fehlerfälle)

### Rolle 4 – GUI & Interaktion

- Konzeption & Umsetzung der GUI
- Anbindung an die Businesslogik
- einfache GUI-Tests oder Testbeschreibung

**Hinweis:** Jede Rolle beinhaltet Programmierarbeit. Es gibt keine reine PM- oder Doku-Rolle.

## 4. Reports

- Reports sind **eigene Komponenten** (keine reine Konsolenausgabe im Controller)
- basieren auf gespeicherten Daten
- deterministisch und testbar
- Ausgabeform frei wählbar:
    - textuell
    - tabellarisch
    - grafisch (z. B. mit `matplotlib.pyplot`, für Lagerstandentwicklung, etc.)

Tests prüfen die **berechneten Werte**, nicht das Layout der Grafik.

## 5. Businesslogik & Datenhaltung

- Datenhaltung ist verpflichtend
- Architektur: **Port-/Adapter-Prinzip**
- **InMemory-Adapter ist Pflicht** (Tests, schneller Start)
- zusätzlicher Persistenz-Adapter frei wählbar (z. B. SQLite, JSON, TinyDB, MongoDB)

Bewertet werden Architektur, Schnittstellentreue und Testbarkeit – nicht das DBMS.

## 6. Versionsverwaltung & Mergekonflikte

- Mehrere Personen arbeiten parallel an gemeinsamen Dateien
- **Mergekonflikte sind erwartet und Teil des Lernziels**
- Bewertungskriterium:
    - Konflikt erkannt (und dokumentiert)
    - verstanden
    - sauber gelöst
    - nachvollziehbarer Commit (→ ebenfalls dokumentiert)

## 7. Versionierung & Changelog

- Gemeinsame Projektversion (Tags im Repo)
- **Jede Person führt ein eigenes Changelog** (`docs/changelog_<name>.md`)
  Dort werden zu jeder Version auch die jeweiligen Commits des Projekts angegeben um den jeweiligen Projektverlauf zu dokumentieren.
- Schnittstellenänderungen zusätzlich in `contracts.md` dokumentieren

Empfohlene Meilensteine:

- v0.1 – Projektstart, Rollen, erste Contracts, Beschreibung der Umsetzung
- v0.2 – Architektur, Walking Skeleton, GUI-Entwurf (Grafik, etc)
- v0.3 – Kernlogik & GUI-Minimum
- v0.4 – erste Reports
- v0.5 – Tests & Stabilisierung
- **v1.0 – fertige, stabile Version**

## 8. Projektmanagement-Dokumente (Abgabe als PDF)

Die folgenden PM-Dokumente sind **als docx zu erstellen und als PDF abzugeben**:

### Inhalt des PM-Dokuments:

1. **Projektcharta**
   - Ziel & Nicht-Ziele
   - Stakeholder
   - Organigramm
   - Risiken
   - Beschreibung der Umsetzung (App via PyQt, cTk oder als gehostete Lösung)
2. **Vorgehensmodell**
   - Kurze Beschreibung (z. B. iterativ / Scrum-light)
   - Ausführliche Begründung der Wahl
3. **Projektstrukturplan (PSP)**
   - strukturierte Gliederung der Projektarbeit
4. **Gantt-Diagramm**
   - zeitliche Planung über die 8 Wochen
   - Bezug zum PSP

## 9. Weitere Projektdokumente im Repository

Pflicht:

- `README.md`
- `docs/contracts.md` (zentral, Rolle 1)
- `docs/architecture.md`
- `docs/tests.md` - `docs/retrospective.md`
- `docs/changelog_<name>.md` (jede Person!)

Optional: - `docs/known_issues.md`

## 10. Zeitplan (8 Wochen)

| Woche | Fokus |
|---|---|
| 1 | Projektstart, Rollen, Repo, Projektcharta |
| 2 | Vorgehensmodell, PSP, Gantt, Schnittstellen |
| 3 | Architektur, Walking Skeleton |
| 4 | Coding Sprint 1 |
| 5 | Coding Sprint 2 + Report A |
| 6 | Report B / Tests |
| 7 | Stabilisierung, Doku |
| 8 | **v1.0**, Abschluss & Präsentation |

**Ziel des Projekts ist nicht maximale Feature-Anzahl, sondern professionelles Vorgehen in Softwareentwicklung und Projektmanagement.**

**WICHTIG:** Die Reihenfolge der Versionen und des Zeitplans stellen eine Leitlinie dar, für all jene die sie brauchen.
Es kann natürlich vorkommen, dass mehrere Mitglieder des Teams zeitgleich fertig werden und man dann eine gemeinsame Version aller Module erstellt. Beurteilt wird somit ein kontinuierlicher Fortschritt im Projekt und nic,0ht die EXAKTE Einhaltung aller Deadlines!

## Beurteilungskonzept des Projekts

Die Beurteilung des Projekts erfolgt **kontinuierlich über den gesamten Projektzeitraum** und nicht ausschließlich anhand der Endabgabe. Ziel ist es, sowohl die **Teamleistung** als auch die **individuelle Leistung** jeder einzelnen Person nachvollziehbar und fair zu bewerten.

Der Schwerpunkt der Bewertung liegt dabei **nicht auf der reinen Programmiermenge**, sondern auf professionellen Arbeitsweisen in der Softwareentwicklung und im Projektmanagement.

**Bewertungsbereiche (Richtwerte)**

- **Projektmanagement & Dokumentation:** ca. **30 %**
  (Projektcharta, Vorgehensmodell, PSP, Gantt, Struktur, Nachvollziehbarkeit)
- **Versionsverwaltung & Zusammenarbeit (Git):** ca. **20 %**
  (Commit-Historie, Branching, Umgang mit Mergekonflikten, individuelle Beiträge)
- **Software, Tests & Reports:** ca. **30 %**
  (Funktionalität, Architektur, Tests, Reports inkl. grafischer Auswertungen)
- **Reflexion & Peer-Feedback:** ca. **20 %**
  (Eigenreflexion, Feedbackkompetenz, Lessons Learned)

**Zwischenabgaben**

Zur Qualitätssicherung und laufenden Rückmeldung gibt es **mehrere Zwischenabgaben**:

1. **Projektstart & Planung (ca. Woche 2)**
   Projektcharta, Vorgehensmodell, PSP, Gantt, Rollenverteilung, erster Schnittstellenentwurf.
2. **Technischer Durchstich (ca. Woche 4)**
   Lauffähiger Grundaufbau (Walking Skeleton), erste GUI-Funktion, Dummy-Daten, erste Tests.
3. **Qualität & Stabilität (ca. Woche 7)**
   Reports, Tests, Changelogs, finale Schnittstellen.

Diese Zwischenabgaben dienen teils der **formativen Rückmeldung**, teils fließen sie **anteilig in die Gesamtnote** ein.

**Peer-Feedback**

Ein kurzes, gezieltes Peer-Feedback ist Teil des Projekts.
Bewertet wird dabei **die Qualität des gegebenen Feedbacks**, nicht das bewertete Team.