KONZEPT

Phase 1: Objektorientierte und funktionale Programmierung mit Python

Peter Wolf
peter.wolf@iu-study.org

Inhaltsverzeichnis

1. Zielsetzung des Projekts

Ziel dieses Projekts ist die prototypische Entwicklung eines Uni-Dashboards zur übersichtlichen Darstellung des Studienfortschritts und zentraler Kennzahlen zur Zielüberwachung, mit Fokus auf Struktur, Nachvollziehbarkeit und objektorientiertem Design.

Das Dashboard soll es ermöglichen,

- studienrelevante Grunddaten (z. B. Studienbeginn, Studiengang) zu erfassen,
- Module anzulegen und absolvierte Module inklusive ECTS und Noten zu verwalten,
- studienbezogene Ziele (z. B. Zielnote, Studiendauer, Arbeitstempo) festzulegen,
- sowie aus diesen Daten automatisch Kennzahlen abzuleiten und darzustellen.

Nicht-Ziele des Projekts sind unter anderem:

- Anbindung an externe Hochschulsysteme,
- Deployment oder produktiver Betrieb,
- Keine visuelle Ausgestaltung, der Fokus der Benutzeroberfläche liegt auf Funktionalität und Struktur

2. Dashboardskizze und angezeigte Inhalte

Die beigefügte Dashboardskizze (PNG) stellt die geplante Benutzeroberfläche auf konzeptioneller Ebene dar.
Sie ist bewusst als funktionale Skizze und nicht als finales UI-Design zu verstehen.

Das Dashboard ist in zwei Hauptbereiche gegliedert:

2.1 Zielübersicht

In der Zielübersicht werden berechnete Kennzahlen angezeigt, die aus den erfassten Studiendaten abgeleitet werden. Dazu gehören unter anderem:

- aktueller Notendurchschnitt,
- Fortschritt Richtung Bachelorabschluss (z. B. Zeit vs. erreichte ECTS),
- aktuelles Arbeitstempo (ECTS pro Monat im Vergleich zum Zielwert).

Die Zielübersicht dient damit primär der Orientierung und Selbsteinschätzung.

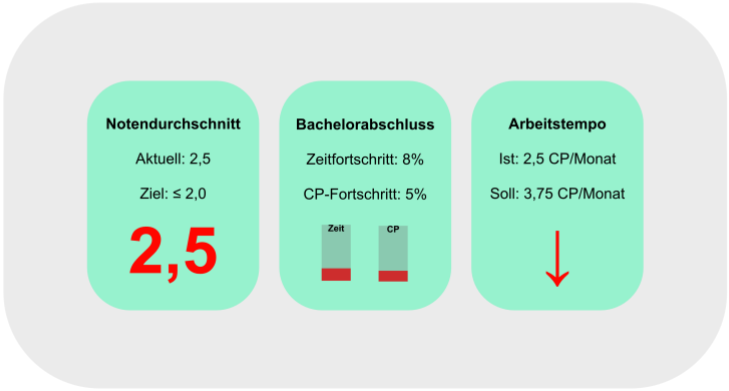
2.2 Datenerfassung

Im Bereich Datenerfassung werden die Daten gepflegt, die als Grundlage für die Auswertungen dienen. Dazu zählen:

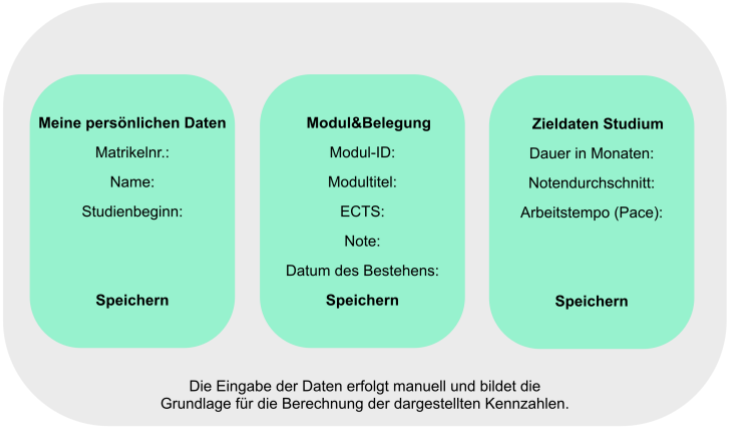
- persönliche Studiendaten (z. B. Studienbeginn),
- Module und deren Belegungen (ECTS, Note, Abschlussdatum),
- Zielparameter wie gewünschte Studiendauer oder Notendurchschnitt.

Die klare Trennung zwischen Dateneingabe und Auswertung war eine bewusste Designentscheidung, da sie sowohl für den Nutzer als auch für die spätere technische Umsetzung eine saubere Struktur vorgibt.

Uni-Dashboard Zielüberwachung



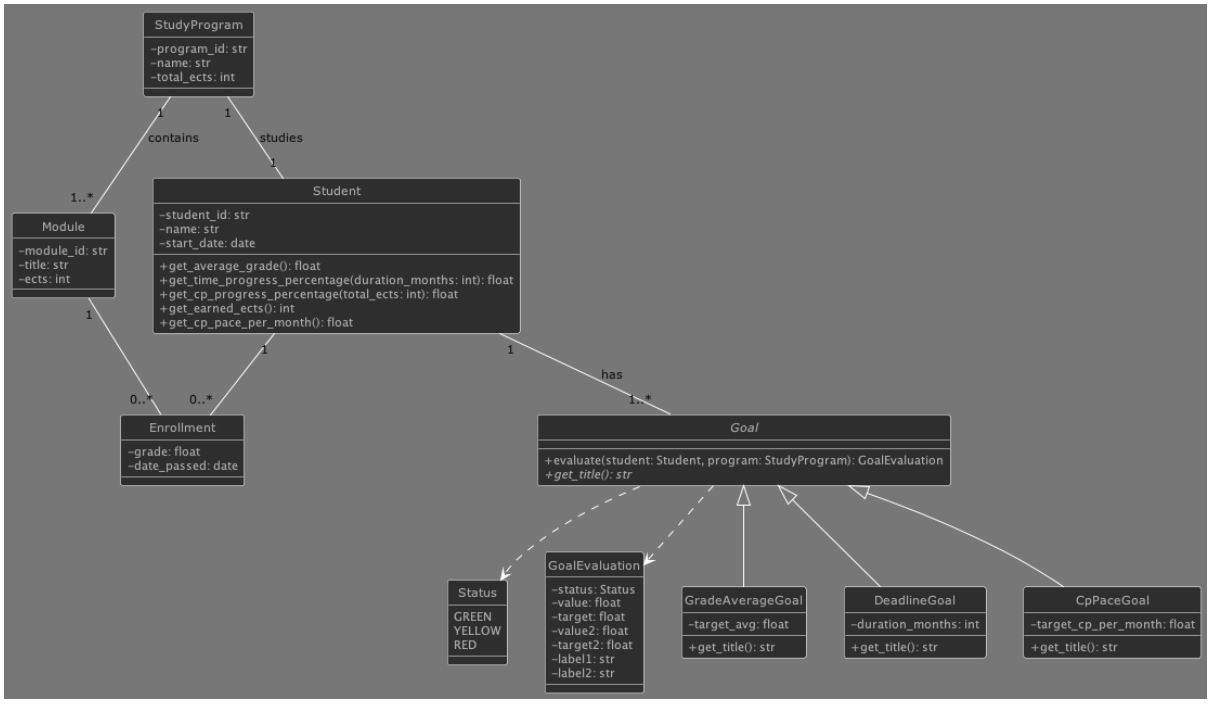
Uni-Dashboard Datenerfassung



3. Entity-UML und Modellierungsentscheidungen

Das Entity-UML (PNG) beschreibt die fachlichen Domänenobjekte des Systems unabhängig von der konkreten technischen Umsetzung.

Zentrale Entitäten sind Student, StudyProgram, Module, Enrollment sowie das Zielkonzept (Goal mit Spezialisierungen und GoalEvaluation).



3.1 Zentrale Modellierungsentscheidungen

- Die Ziele wurden als **abstrakte Basisklasse (Goal)** mit konkreten Spezialisierungen modelliert.
  Dadurch lassen sich neue Zielarten später hinzufügen, ohne bestehende Logik zu verändern (Open-Closed-Prinzip).
- Die Beziehung zwischen Student, Module und Enrollment wurde bewusst explizit modelliert, um reale Zusammenhänge (mehrere Module, unterschiedliche Abschlusszeitpunkte) korrekt abzubilden.
- Auf eine eigene Semester-Entität wurde bewusst verzichtet, da Noten ausschließlich für tatsächlich belegte Module erfasst werden und eine formale Semesterzuordnung keinen zusätzlichen funktionalen Mehrwert bietet.

3.2 Kritische Diskussion

Alternativ wäre ein vereinfachtes Modell mit weniger Klassen möglich gewesen, was den initialen Implementierungsaufwand reduziert hätte, jedoch zu einer stärkeren Vermischung von Verantwortlichkeiten geführt hätte.

Die gewählte Modellierung ist daher bewusst strukturierter gehalten, um objektorientierte Prinzipien klar darzustellen und Erweiterungen konzeptionell offen zu halten.

4. Geplante Programmierumgebung und Werkzeuge

Für die Umsetzung des Prototyps wurde folgende Umgebung gewählt:

4.1 Programmiersprache

- **Python 3.12**
  Python wurde gewählt, da diese Programmiersprache zum einen Bestandteil meines Studiums ist und zum anderen, da es eine klare Syntax, gute Lesbarkeit, umfangreiche Standardbibliotheken bietet und bereits auf meinem MacBook Pro installiert war, da ich unter Anderem in Python schon ein paar kleine Projekte umgesetzt habe.
  Python 3.12 ist stabil und eine aktuelle Version, die bereits im produktiven Einsatz erprobt ist.

4.2 Entwicklungsumgebung

- **Visual Studio Code**
  VS-Code bietet eine gute Python-Integration, Debugging-Möglichkeiten und eine übersichtliche Projektstruktur.

4.3 Bibliotheken und Technologien

- **tkinter**
  Wird für die grafische Benutzeroberfläche verwendet und ist für einen Desktop-Prototypen ausreichend.
- **sqlite3**
  SQLite als lokale Datenbank ist leichtgewichtig, benötigt keinen separaten Server und eignet sich daher gut für eine prototypische Umsetzung.
- **datetime**
  Wird zur Verarbeitung von Datumswerten verwendet.
- **logging**
  Wird eingesetzt, um wichtige Programmabläufe und Fehler nachvollziehbar zu protokollieren.
- **dataclasses**
  Wird verwendet, um Datenobjekte kompakt und gut lesbar zu definieren. Dadurch wird Boilerplate reduziert, ohne die Objektorientierung zu verwässern.

- **abc**
  Wird genutzt, um abstrakte Basisklassen zu definieren (z. B. für das Zielkonzept Goal). So lässt sich festlegen, welche Methoden jede Zielklasse implementieren muss.
- **enum**
  Wird für feste Wertemengen eingesetzt, z. B. zur Darstellung eines Status (grün/gelb/rot) bei Zielbewertungen.
- **typing**
  Wird verwendet, um Methodenschnittstellen klarer zu beschreiben (z. B. Rückgabewerte wie „Objekt oder None“, Listen von Objekten).

4.4 Begründung der Auswahl

Alle verwendeten Werkzeuge und Bibliotheken sind entweder Bestandteil der Standardbibliothek oder weit verbreitet.
Dies reduziert technische Risiken und stellt sicher, dass der Fokus des Projekts auf Konzept, Struktur und OOP-Prinzipien liegt und nicht auf der Konfiguration externer Abhängigkeiten.

Abschlussbemerkung

Im Rahmen der Modellierung habe ich versucht, grundlegende objektorientierte Prinzipien – insbesondere aus dem SOLID-Umfeld – zu berücksichtigen.
Dies zeigt sich vor allem in der klaren Trennung von Verantwortlichkeiten sowie in der Erweiterbarkeit des Zielkonzepts, ohne bestehende Klassen anpassen zu müssen.

Die prototypische Umsetzung soll verdeutlichen, dass das entworfene Konzept objektorientiert, strukturiert und technisch umsetzbar ist.
Die in Phase 1 erarbeiteten Artefakte (Dashboard-Skizze, UML, Architekturüberlegungen) bilden die Grundlage für die anschließende Erarbeitungs- und Reflexionsphase.