Objektorientierte und funktionale Programmierung mit Python (DLBDSOOFPP01_D)
Aufgabenstellung: Studien-Dashboard (OOP Python)
Name: Gökhan Sen / Matrikelnummer: IU14140216

---

**Phase 3 - Abstract – Studien-Dashboard**

Im Rahmen dieses Projekts wurde ein objektorientiertes Studien-Dashboard in Python konzipiert, modelliert und implementiert. Ziel war die Entwicklung einer lauffähigen Anwendung, die den individuellen Studienverlauf strukturiert erfasst und zentrale Leistungskennzahlen (KPIs) visualisiert. Der gesamte Entwicklungsprozess – von der fachlichen Konzeption über die UML-Modellierung bis zur implementierten Software – wurde nachvollziehbar dokumentiert.

Ausgangspunkt war die fachliche Modellierung der relevanten Entitäten (Student, Studiengang, Modul, Modulbelegung und Semester) sowie die Definition geeigneter Kennzahlen. Als zentrale KPIs wurden der Studienfortschritt in ECTS, die gewichtete Durchschnittsnote und deren zeitlicher Verlauf festgelegt. Bereits in der Konzeptionsphase zeigte sich, dass die präzise Definition dieser Kennzahlen entscheidend für die spätere Architektur ist, da sie sowohl Datenmodell als auch Service-Logik beeinflusst.

In Phase 2 wurde die Gesamtarchitektur in UML modelliert und eine Schichtenstruktur definiert, die Benutzeroberfläche, Geschäftslogik und Persistenz klar voneinander trennt. In Phase 3 wurde diese Architektur implementiert und bewusst weiterentwickelt. Gegenüber der Phase-2-Modellierung wurden einzelne Strukturen konsolidiert: Die ursprünglich getrennten Komponenten für CRUD-Operationen und Visualisierung wurden in einem zentralen DashboardService gebündelt, konzeptionelle UI-Elemente zugunsten einer schlankeren Tkinter-Implementierung reduziert. Die Visualisierungslogik wird nun über klar definierte Service-Methoden bereitgestellt.

Diese Anpassungen erfolgten ohne Veränderung der fachlichen Anforderungen und dienten einer gezielten architektonischen Verdichtung. Durch die Zusammenführung der Geschäftslogik in einer zentralen Service-Schicht wurden Kohärenz und Wartbarkeit verbessert. Gleichzeitig bleibt die Datenzugriffsschicht über Repository-Klassen klar gekapselt, wodurch die Anwendung strukturell stabil und erweiterbar bleibt.

Die gewählte Schichtenarchitektur (UI → Service → Repository → Datenbank/Protokoll) trennt Darstellung, Geschäftslogik und Persistenz strikt. Dadurch ist die KPI-Logik unabhängig von der konkreten Benutzeroberfläche oder Datenbankimplementierung; Änderungen betreffen jeweils nur die entsprechende Schicht. Dies erhöht die Testbarkeit der Geschäftslogik, verbessert die Verständlichkeit des Codes und erleichtert zukünftige Erweiterungen.

Eine wesentliche Erkenntnis war, dass ein konzeptionell sauberes Domänenmodell zwar essenziell ist, in der Implementierung jedoch pragmatisch angepasst werden muss. So wurde auf eine eigenständige Semester-Entität verzichtet und stattdessen mit Attributen in Modul- bzw. Belegungsobjekten gearbeitet, um die Persistenzstruktur zu vereinfachen. Diese Entscheidung reduzierte Komplexität ohne funktionale Einbußen.

Im Projektverlauf zeigte sich zudem, dass insbesondere die präzise Definition und Berechnung der KPIs mehr Iterationen erforderte als zunächst angenommen. Darüber hinaus wurde bewusst auf automatisierte Tests sowie auf Import-/Export-Funktionen verzichtet, um den Projektumfang im vorgesehenen Rahmen zu halten. Die Anwendung ist als Single-User-Demonstrator konzipiert und nicht für parallele Mehrbenutzer-Szenarien ausgelegt. Diese Aspekte stellen Ansatzpunkte für eine zukünftige Weiterentwicklung dar.

Besonders positiv hervorzuheben ist die durchgängige Nachvollziehbarkeit von der UML-Modellierung bis zur implementierten Klassenstruktur. Die im finalen Klassendiagramm dargestellten Komponenten entsprechen der realisierten Codebasis und verdeutlichen die konsequente Anwendung objektorientierter Prinzipien wie Kapselung, klare Verantwortlichkeiten und lose Kopplung zwischen den Schichten.

Zusammenfassend zeigt das Projekt die vollständige Umsetzung eines objektorientierten Softwareentwicklungsprozesses – von der fachlichen Analyse über die architektonische Modellierung bis zur lauffähigen Anwendung. Es dokumentiert bewusst getroffene Designentscheidungen sowie deren Auswirkungen auf Wartbarkeit, Erweiterbarkeit und Verständlichkeit.