

Abschlussprüfung Winter 2025/2026 Fachinformatiker für Anwendungsentwicklung Dokumentation zur betrieblichen Projektarbeit

Entwicklung eine „Interaktive UML-Lern-App mit Quiz, Notizen, Lernpfaden (Levels) und Kursverwaltung“

25. September 2025

Marburg, 2025

Eingereicht von:

Josiane Kanouo Maneyo

Umschülerin zur Fachinformatikerin für Anwendungsentwicklung



IAD GmbH Marburg

Neue Kasseler Straße 62e

35039 Marburg

Betreuer: Marcus Brauer

Abgabedatum:

Inhaltsverzeichnis

Abbildungsverzeichnis III

Tabellenverzeichnis IV

Listings V

Abkürzungsverzeichnis VI

Inhalt

[1 Einleitung 1](#_Toc210643342)

[1.1 Projektumfeld 1](#_Toc210643343)

[1.2 Projektziel 1](#_Toc210643344)

[1.3 Projektbegründung 2](#_Toc210643345)

[1.4 Projektschnittstellen 2](#_Toc210643346)

[1.5 Projektabgrenzung 2](#_Toc210643347)

[2 Projektplanung 2](#_Toc210643348)

[2.1 Projektphasen 2](#_Toc210643349)

[2.2 Abweichungen vom Antrag 2](#_Toc210643350)

[2.3 Ressourcenplanung 2](#_Toc210643351)

[2.4 Vorgehensmodell 3](#_Toc210643352)

# Einleitung

Dieses Kapitel beschreibt die Ausgangssituation, die Motivation und die Zielsetzung des Projekts. Es erläutert, welches Problem durch die Anwendung gelöst werden soll, welchen Nutzen sie bietet und wie das Projekt in den Gesamtkontext der Ausbildung eingebettet ist. Darüber hinaus werden die Projektumgebung, die Beteiligten und die Rahmenbedingungen vorgestellt, um einen Überblick über den Hintergrund des Projekts zu geben.

## Projektumfeld

Das Projekt wird im Rahmen der Umschulung zur Fachinformatikerin für Anwendungsentwicklung an der IAD Bildungszentrum Marburg GmbH durchgeführt. Die IAD, Teil der IAD Group mit über 50 Jahren Erfahrung, ist einer der größten privaten Bildungsanbieter in Deutschland für IT und Management. Sie bietet praxisorientierte Ausbildungen und Umschulungen an, darunter den dualen Fachinformatiker für Anwendungsentwicklung mit Fokus auf Softwareentwicklung (z. B. Kotlin, Mobile Apps) und Datenhaltung. Der Marburg-Standort (Wilhelm-Röpke-Straße 11) dient als zentraler Hub mit modernen IT-Labs und Kooperationen mit der Agentur für Arbeit. Jährlich werden ca. 8.000 Teilnehmer bundesweit qualifiziert, mit hohen Erfolgsquoten: 90 % der Absolventen finden direkt einen Job, 95 % Zufriedenheit und 98 % Bestehensrate bei Prüfungen. Die Projektarbeit erfolgt als Einzelprojekt. Die Verantwortung umfasst Analyse, Entwurf, Implementierung, Test und Dokumentation. Fachliche Betreuung erfolgt durch Marcus Brauer, Dozent an der IAD, der die technische Umsetzung unterstützt und die Einhaltung der IHK-Vorgaben überwacht. Zielgruppe der Anwendung sind Lernende und Lehrkräfte, die eine digitale Plattform zur Kursauswahl, Bearbeitung von Quizfragen, Verwaltung von Fehlerlisten und Erstellung persönlicher Notizen nutzen können. Die Anwendung adressiert den Übergang von statischen Papierübungen zu interaktiven Tools, die ortsunabhängiges Training mit direktem Feedback und zentraler Auswertung ermöglichen. Neben der bestehenden Lernumgebung wird mit der App eine moderne, mobile Erweiterung geschaffen, die neuen Lernenden den Einstieg durch ein interaktives Tutorial erleichtert und Rückmeldungen direkt in die Weiterentwicklung einfließen lässt.

## Projektziel

Ziel des Projekts ist die Entwicklung einer interaktiven Android-Anwendung, die Lernenden und Lehrkräften des IAD Bildungszentrums ein Werkzeug zum Üben, Wiederholen und Verwalten von Lerninhalten bietet. Die Anwendung ersetzt traditionelle, statische Methoden und schafft eine strukturierte, motivierende Lernumgebung für UML-Diagramme.

Die App umfasst Funktionen wie Kursauswahl, Quiz mit direktem Feedback, Fehlerlisten, persönliche Notizen, ein Levelsystem und einen Prüfungsmodus mit Timer. Administratoren erhalten einen Bereich zur Verwaltung von Kursen und Fragen. Technisch wird die Anwendung mit Kotlin in Android Studio umgesetzt und nutzt Firebase für Authentifizierung, Datenspeicherung in Firestore und Medienverwaltung in Storage. Sicherheitsregeln und Offline-Funktionen gewährleisten Stabilität auch ohne Internetverbindung, wodurch Lernende von transparenter Fortschrittskontrolle profitieren und Lehrkräfte zentrale Inhaltsverwaltung erhalten. Neben den Kernfunktionen wird die App um zwei wesentliche Erweiterungen ergänzt: ein Onboarding-Tutorial, das neuen Nutzenden in wenigen Schritten die wichtigsten Funktionen erklärt, und eine In-App-Feedback-Funktion, mit der Rückmeldungen direkt an die Entwicklerin gesendet werden können. Dadurch wird der Einstieg erleichtert, die Benutzerfreundlichkeit erhöht und die kontinuierliche Verbesserung der Anwendung unterstützt.

## Projektbegründung

Der Unterricht an der IAD basiert derzeit überwiegend auf Papierübungen und Word-Dokumenten für UML-Diagramme. Diese Methode ist unflexibel, bietet kein direktes Feedback und erlaubt keine zentrale Verwaltung oder Auswertung. Lehrkräfte haben dadurch keinen einfachen Überblick über den Lernstand, und Ergebnisse lassen sich nicht systematisch dokumentieren oder vergleichen. Die geplante Anwendung digitalisiert diese Prozesse, erhöht Transparenz und steigert die Motivation. Lernende können gezielt Fehler wiederholen, ihren Fortschritt verfolgen und individuelle Notizen anlegen.

Eine Eigenentwicklung wurde gewählt, da bestehende Lernplattformen wie Moodle oder Quizlet keine Kombination aus Fehlerwiederholung, Notizen und Levelsystem bieten. Die Lösung ist somit speziell auf die Bedürfnisse der Umschulungsteilnehmenden zugeschnitten und unterstützt die Digitalisierung des Unterrichts, wodurch der Lernprozess effizienter, praxisnäher und motivierender gestaltet wird als mit herkömmlichen Papierübungen. Durch das Onboarding-Tutorial wird der Einstieg in die Anwendung auch für technisch weniger erfahrene Lernende deutlich einfacher. Die integrierte Feedback-Funktion ermöglicht es, Verbesserungsvorschläge und Fehler direkt zu melden, was eine nachhaltige Weiterentwicklung und Qualitätssicherung unterstützten.

## Projektschnittstellen

Die Anwendung kommuniziert ausschließlich mit Firebase-Diensten (Authentication, Cloud Firestore, Storage, Security Rules) über sichere HTTPS-Verbindungen. Offline-Nutzung wird durch lokale Datenspeicherung ermöglicht. Beteiligte Personen sind Marcus Brauer als Auftraggeber, Josiane Kanouo Maneyo als Entwicklerin und Testpersonen aus der IAD. Externe Systeme oder APIs werden nicht integriert. Zusätzlich wird die Feedback-Sammlung in Firebase zur Speicherung eingehender Rückmeldungen genutzt.

## 1.5 Projektabgrenzung

Das Projekt beschränkt sich auf die Entwicklung der Android-Anwendung mit den im Antrag festgelegten Kernfunktionen. Ausgeschlossen sind eine Webversion, KI-Funktionen, komplexe Statistiken oder die Veröffentlichung im Google Play Store. Das Projekt endet mit einem funktionsfähigen Prototyp, der die geforderten Anforderungen erfüllt. Es wird keine UML-Diagramm-Erstellung integriert; der Fokus liegt auf quizbasiertem Lernen und Verwaltung.

# Projektplanung

Im folgenden Kapitel wird der zeitliche und organisatorische Ablauf des Projekts dargestellt. Hierzu werden die einzelnen Projektphasen beschrieben, der Zeitaufwand dokumentiert und die eingesetzten Methoden und Werkzeuge erläutert.

## 2.1 Projektphasen

Das Projekt umfasst einen Zeitrahmen von 80 Stunden und gliedert sich in acht aufeinanderfolgende Phasen: Analyse, Konzept, Entwurf, Implementierung, Test und Dokumentation. Die Analysephase diente der Anforderungsermittlung, gefolgt von Entwurf und Programmierung sowie Tests zur Sicherstellung der Funktionalität. Die Soll-Phase und Implementierung umfassen zusätzlich die Entwicklung eines Onboarding-Tutorials sowie einer In-App-Feedback-Funktion, um die Benutzerfreundlichkeit zu erhöhen. Eine detaillierte Übersicht der Stundenverteilung ist im Anhang A1 enthalten.

## 2.2 Abweichungen vom Antrag

Gegenüber dem ursprünglichen Antrag wurde der Funktionsumfang leicht erweitert.  
Ergänzt wurden ein Onboarding-Tutorial und eine In-App-Feedback-Funktion, um das Nutzererlebnis zu verbessern. Der Zeitrahmen von 80 Stunden blieb unverändert; der Mehraufwand wurde durch optimierte Arbeitsschritte ausgeglichen.

## 2.3 Ressourcenplanung

Zur Umsetzung stehen ein Windows-Laptop, ein Android-Smartphone sowie die Software Android Studio, Kotlin und Firebase zur Verfügung. Versionierung und Datensicherung erfolgen über GitHub. Zusätzliche Tools wie Draw.io, Microsoft Word und Latex unterstützen die Erstellung von Diagrammen und der Dokumentation.

An Personalressourcen wurden sieben Stunden für einen fachlichen Betreuer einkalkuliert, der bei fachlichen Themen unterstützen konnte und Teile der Analyse- sowie Planungsphase begleitete. Außerdem wurden zwei Stunden für einen technischen Betreuer eingeplant, der bei technischen Themen unterstützen und Feedback zu der Entwicklung geben konnte.

## 2.4 Vorgehensmodell

Es wurde ein phasenorientiertes Vorgehensmodell mit agilen Elementen angewandt. Die Struktur orientiert sich am Wasserfallmodell, ergänzt durch kurze Feedbackzyklen während der Implementierung. Dies ermöglichte eine flexible Reaktion auf Designänderungen und Testergebnisse bei gleichzeitiger Einhaltung eines klaren Ablaufs. Während der Implementierung wurden nach jeder abgeschlossenen Funktion (Login, Quiz, Feedback, Tutorial) kurze Test- und Feedback-Zyklen durchgeführt, um Stabilität und Nutzerfreundlichkeit sicherzustellen.

# 3 Analysephase

In der Analysephase werden die Anforderungen an die zu entwickelnde Anwendung systematisch erfasst und präzisiert. Dabei werden sowohl funktionale als auch nicht-funktionale Anforderungen beschrieben, um sicherzustellen, dass alle relevanten Aspekte berücksichtigt werden. Zusätzlich wird das bestehende Problem analysiert, und es werden die Ziele sowie der Nutzen der geplanten Lösung klar abgegrenzt. Diese Phase bildet die inhaltliche Grundlage für den späteren Entwurf und die technische Umsetzung.

## 3.1 Ist-Analyse

Der Unterricht an der IAD Bildungszentrum Marburg GmbH basiert derzeit auf klassischen Lehrmethoden, bei denen die Lernenden UML-Diagramme in Papier- oder PDF-Form üben. Eine unmittelbare Rückmeldung zu den Ergebnissen erfolgt nicht; Korrekturen werden von Lehrkräften manuell durchgeführt und können nicht zentral dokumentiert werden. Auch der individuelle Lernfortschritt ist nicht transparent einsehbar. Vorhandene Tools wie draw.io oder PlantUML² ermöglichen zwar das Zeichnen von Diagrammen, unterstützen jedoch keinen interaktiven Lernprozess mit Feedback-, Notiz- oder Fortschrittsfunktionen. Das Fehlen einer digitalen, benutzerfreundlichen Lösung führt zu hohem Zeitaufwand und geringerer Motivation bei den Lernenden. Eine moderne, mobile App-Lösung kann diese Probleme beseitigen, indem sie Lernfortschritte automatisch erfasst, Fehler dokumentiert und Lerninhalte flexibel zur Verfügung stellt.

## 3.2 **Funktionale Anforderungen**

## Ziel des Projekts ist die Entwicklung einer interaktiven Android-App, die den Lernprozess für UML Diagramme digitalisiert und Lernende beim eigenständigen Üben unterstützt. Die Anwendung bietet einen rollenbasierten Zugang über Firebase Authentication, bei dem zwischen *User* (Lernende) und *Admin* (Lehrkräfte) unterschieden wird. Lernende können sich registrieren, anmelden, Kurse auswählen, Quizze durchführen, Notizen anlegen und Feedback abgeben. Administratoren verwalten Kurse, Lerneinheiten und Fragen direkt in der App. Alle Benutzerdaten und Lernfortschritte werden in Firebase Firestore gespeichert und beim Login automatisch der jeweiligen Rolle zugeordnet. Die App umfasst folgende Hauptfunktionen: Ein Kurs- und Lerneinheitensystem strukturiert die Inhalte in Themenbereiche (*Courses* und *Units*), die jeweils Quizfragen mit Antwortoptionen, Lösung und Punktewert enthalten. Das Quizmodul lädt die Fragen dynamisch aus Firestore, zeigt sie nacheinander an und berechnet nach Abschluss Punktestand, Erfolgsquote und Level. Eine Fehlerliste **F**speichert falsch beantwortete Fragen für gezieltes Wiederholen, während ein integrierter Timer die Bearbeitungszeit begrenzt. Über die Notizenfunktion können Lernende eigene Gedanken oder Erklärungen speichern, die sowohl offline verfügbar als auch automatisch synchronisiert werden. Das Level- und Punktesystem motiviert durch Fortschrittsanzeigen und Levelaufstiege, die im Profil dargestellt werden. Zur Unterstützung des Lernprozesses enthält die App ein kurzes Onboarding-Tutorial, das neue Nutzer durch die wichtigsten Bereiche führt, sowie eine Feedback-Funktion, über die Anregungen oder Fehlermeldungen direkt an die Entwicklerin gesendet werden können. Ein separater Administrationsbereich erlaubt Lehrkräften das Erstellen und Bearbeiten von Kursen und Fragen in Echtzeit. Die gesamte Kommunikation erfolgt verschlüsselt, und die Firestore-Regeln sichern die Daten gegen unbefugten Zugriff. Dank Offline-Caching bleibt die Anwendung auch ohne Internetverbindung voll funktionsfähig. Damit erfüllt die UML-Lern-App sämtliche im Projektantrag definierten Anforderungen. Eine detaillierte tabellarische Funktionsübersicht mit technischen Zuordnungen ist im Anhang A2 enthalten.

## 3.3 Wirtschaftlichkeitsbetrachtung

## Die Wirtschaftlichkeit des Projekts wurde anhand der Entwicklungskosten und des erwarteten Nutzens für die IAD untersucht. Da die Entwicklerin als Umschülerin kein Gehalt erhält, entfallen Personalkosten. Stattdessen wurde der Zeitaufwand der betreuenden Fachkraft als kalkulatorischer Aufwand berücksichtigt. Ziel der Betrachtung war es, den Nutzen für die IAD darzustellen, insbesondere die Zeitersparnis für Dozenten und die Reduktion des Verwaltungsaufwands durch die App.

## 3.3.1 Make-or-Buy-Entscheidung

## Zu Beginn wurde geprüft, ob eine bestehende Lernplattform wie Moodle, OpenOLAT oder Udemy Business eingesetzt werden könnte. Diese Systeme bieten zwar umfangreiche Funktionen, sind jedoch serverbasiert, kostenpflichtig und nicht auf Offline-Nutzung ausgelegt. Zudem wäre eine Anpassung an die spezifischen Anforderungen der IAD – insbesondere das rollenbasierte Rechtesystem (Trainer/Lernende) und die Integration von Firebase mit zusätzlichem Aufwand verbunden.

## Die Eigenentwicklung der UML-Lern-App in Kotlin mit Firebase Firestore bot dagegen folgende Vorteile: Keine Lizenz- oder WartungskostenL; Offline-Fähigkeit; volle Kontrolle über Datenspeicherung (DSGVO-konform, Firebase Cloud in EU-Region); Erweiterbarkeit für künftige; Module oder Kurse . Auf Basis dieser Argumente fiel die Entscheidung auf die Eigenentwicklung („Make“).

### 3.3.2 Projektkosten

## Da keine Lohnkosten für die Entwicklerin anfallen, beschränken sich die Projektkosten auf den Betreuungsaufwand, Arbeitsmittel und Infrastruktur. Die Kalkulation basiert auf einem Ansatz von 8 Stunden für den Betreuer (technische Begleitung, Feedback, Abnahme) sowie den pauschalen Nutzungskosten der Hardware und Firebase-Umgebung.

## Kostenart Anzahl / Zeit Satz Kosten (€) Bemerkung

## Betreueraufwand (IAD) 8 Stunden 40 € / Std. 320 € Fachliche Betreuung, Abnahme

## Arbeitsplatz & Internet 80 Stunden 5 € / Std. 400 € Strom, PC, Verbindung

## Firebase-Ressourcen 3 Monate 10 € / Monat 30 € Cloud-Nutzung (Speicher, Auth)

## Gesamtkosten ≈ 750 €

## Da der Firebase Spark Plan (kostenlose Stufe) genutzt wurde, fallen keine laufenden Kosten an.

## Eventuelle Erweiterungen (z. B. Analytics oder Cloud Functions) könnten in der Zukunft geringe Zusatzkosten verursachen.

### 3.3.3 Nutzen und Einsparungen

## Die App reduziert den administrativen Aufwand erheblich. Vor der Einführung mussten Trainer Quizfragen ausdrucken, manuell korrigieren und Ergebnisse übertragen. Pro Kurs mit 20 Lernenden dauerte dies durchschnittlich 10 Minuten pro Lernenden und Woche. Durch die automatische Auswertung in der App entfallen diese Aufgaben fast vollständig. Bei 20 Lernenden über 40 Unterrichtswochen ergibt sich:

## 20 Lernende × 10 Minuten × 40 Wochen = 8 000 Minuten = 133 Stunden pro Jahr

## Bei einem Dozentenstundensatz von 30 € entspricht das einer jährlichen Einsparung von:

## 133 Stunden × 30 € = ≈ 4 000 €

## Weitere qualitative Vorteile:

## Trainer können die gewonnene Zeit für inhaltliche Betreuung oder neue Kurse nutzen.

## Lernende erhalten sofortiges Feedback, was den Lernerfolg verbessert.

## Papierverbrauch und Druckkosten sinken.

## Ergebnisse sind digital archiviert und jederzeit auswertbar.

## Das Image der IAD als moderne Bildungseinrichtung wird gestärkt.

### 3.3.4 Amortisation und Fazit

## Die Gesamtkosten der Entwicklung belaufen sich auf rund 750 €.

## Bei einer jährlichen Zeitersparnis von etwa 4 000 € amortisiert sich die App bereits nach:

## 750 € / 4 000 € ≈ 0,19 Jahre = ca. 2,3 Monate

## Damit ist das Projekt nicht nur wirtschaftlich sinnvoll, sondern auch langfristig profitabel.

## Die Anwendung spart Dozenten dauerhaft Zeit, senkt interne Aufwände und verbessert die Unterrichtsqualität. Auch wenn keine direkten Einnahmen entstehen, führt der indirekte Nutzen – Zeitgewinn, Qualitätssteigerung, Digitalisierungsimage – zu einer messbaren Effizienzsteigerung für die IAD. Das Projekt trägt somit unmittelbar zur wirtschaftlichen und organisatorischen Optimierung der Schulungsprozesse bei.

## 3.4 Qualitätsanforderungen

## Neben den fachlichen Funktionen wurden auch qualitative Anforderungen definiert, die Leistung, Sicherheit und Benutzerfreundlichkeit der Anwendung sicherstellen. Die App soll stabil, intuitiv und plattformgerecht funktionieren und dabei den Datenschutzbestimmungen entsprechen. Die Benutzerfreundlichkeit steht im Vordergrund: Eine klare, übersichtliche Oberfläche nach den Richtlinien des Material Design sorgt für eine intuitive Bedienung. Alle Texte und Icons sind für mobile Geräte optimiert, und Animationen unterstützen den Lernfluss, ohne abzulenken. Zur Leistung zählt, dass das Quiz, auch bei mehreren Datenabfragen, schnell reagiert und Ergebnisse in Echtzeit anzeigt. Durch lokale Zwischenspeicherung (Offline-Cache) bleibt die App auch bei instabiler Verbindung funktionsfähig, wodurch eine hohe Verfügbarkeit erreicht wird. Hinsichtlich Sicherheit werden alle Benutzerdaten ausschließlich über verschlüsselte HTTPS-Verbindungen übertragen. Die Authentifizierung erfolgt über Firebase, sodass keine sensiblen Passwörter in der App gespeichert werden. Über Firestore Security Rules wird der Zugriff auf Daten strikt nach Benutzerrolle kontrolliert. Die App erfüllt zudem grundlegende Anforderungen an Datenschutz und Integrität, indem nur notwendige personenbezogene Daten (z. B. Benutzername, UID) gespeichert werden.Ein weiteres Ziel war die Wartbarkeit und Erweiterbarkeit. Durch den modularen Aufbau im MVVM-Architekturmuster (Model–View–ViewModel) können neue Funktionen wie zusätzliche Kurse, Auswertungen oder Lernstatistiken ohne größere Codeanpassungen ergänzt werden.

## Die App ist so ausgelegt, dass sie langfristig um weitere Themengebiete oder Quiztypen erweitert werden kann.

## 3.5 Anwendungsfälle

Das System unterstützt zwei Akteure: Lernende und Administratoren. Lernende können nach dem Login Kurse auswählen, Fragen beantworten und Ergebnisse direkt einsehen. Fehler werden gespeichert und können wiederholt geübt werden. Administratoren verwenden die App, um Kurse und Fragen zu pflegen und Inhalte laufend zu aktualisieren. Das Onboarding-Tutorial erleichtert den Einstieg für neue Nutzende, während die Feedback-Funktion Rückmeldungen an die Entwicklerin ermöglicht. Eine grafische Darstellung der Anwendungsfälle befindet sich im Anhang A3.

## 3.6 Verweis auf das Lastenheft

Alle funktionalen und nicht-funktionalen Anforderungen sind im Anhang A4 – Lastenheft detailliert beschrieben. Das Lastenheft definiert Muss- und Kann-Kriterien sowie Abgrenzungen des Projekts und bildet die Grundlage für die Abnahme.

# 4 Entw4urfsphase

Dieses Kapitel beschreibt die technische und gestalterische Planung der UML-Lern-App. Hier werden die grundlegende Architektur, die Struktur der Datenbank, das Design der Benutzeroberfläche sowie der Aufbau der App erläutert. Die Entwurfsphase bildet somit den Übergang zwischen der theoretischen Planung und der praktischen Umsetzung.

## Zielplattform

Für die Umsetzung der UML-Lern-App wurde die Android-Plattform als Zielumgebung gewählt. Ausschlaggebend dafür waren die weite Verbreitung von Android-Geräten, die einfache Verfügbarkeit von Entwicklungswerkzeugen sowie die Möglichkeit, moderne Technologien wie Firebase direkt zu integrieren. Als Programmiersprache kommt Kotlin (Version 1.9) zum Einsatz, da sie von Google offiziell unterstützt wird, eine klare Syntax besitzt und sich besonders gut für mobile Anwendungen eignet. Die Datenhaltung erfolgt in der Cloud Firestore-Datenbank von Firebase, die eine dokumentenbasierte Struktur bietet und sowohl Online- als auch Offline-Betrieb ermöglicht. Diese Architektur erlaubt eine flexible, sichere und leicht erweiterbare Speicherung aller Benutzerdaten, Kurse und Quiz-Ergebnisse. Die App folgt dem Client-Server-Prinzip: Der Client (die Android-App) kommuniziert mit dem Server (Firebase-Dienste) über eine gesicherte HTTPS-Verbindung. Dadurch können Benutzer weltweit auf ihre persönlichen Daten zugreifen, ohne dass lokale Server notwendig sind. Die Anwendung wurde auf Emulatoren (Pixel 6, Android 14) und auf realen Geräten wie dem Samsung A56 getestet. Die Entwicklungsumgebung war über ein GitHub-Repository versioniert, sodass alle Änderungen nachvollziehbar dokumentiert wurden. Damit wird sichergestellt, dass die Anwendung flüssig läuft und gleichzeitig auch auf älteren Geräten nutzbar bleibt. Die Kombination aus Kotlin, Firebase und Android Studio bietet somit eine moderne, performante und zukunftssichere Grundlage für die Umsetzung der Lern-App.

## 4.2 Architektur und Designprinzip

Die Architektur der UML-Lern-App basiert auf dem Model-View-ViewModel-Muster (MVVM), einem bewährten Architekturkonzept für Android-Anwendungen. Dieses Muster trennt die Datenhaltung, die Geschäftslogik und die Benutzeroberfläche klar voneinander. Dadurch bleibt die Anwendung modular aufgebaut, leicht wartbar und flexibel erweiterbar.

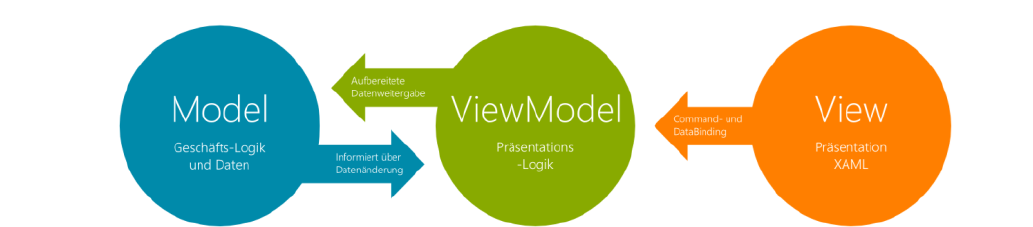


Abbildung 1 MVVM- Prinzip

Das Model repräsentiert die Datenebene. Hier befinden sich alle Datenklassen, die direkt der Struktur der Firestore-Datenbank entsprechen – beispielsweise *User*, *Admin*, *Course*, *Unit*, *Question*, *Attempt*, *Note* und *Feedback*. Diese Klassen enthalten keine Logik, sondern dienen ausschließlich der Datenhaltung und -übertragung.

Das ViewModel bildet die Vermittlungsschicht zwischen Daten und Benutzeroberfläche. Es ruft Daten aus dem Repository ab, validiert Eingaben, verarbeitet sie und bereitet sie für die Anzeige in der View vor. Außerdem überwacht das ViewModel Änderungen und aktualisiert die Oberfläche automatisch über LiveData, sobald sich Daten in Firestore ändern. Die View umfasst die Benutzeroberfläche – also Activities, Fragments und die zugehörigen XML-Layouts. Sie ist für die Darstellung und Interaktion zuständig, enthält jedoch keine Geschäftslogik. Über Data Binding kommuniziert sie direkt mit dem ViewModel, wodurch eine klare Trennung zwischen Darstellung und Logik gewährleistet ist. Zwischen ViewModel und Datenebene vermittelt ein Repository, dass alle Firebase-Operationen (Create, Read, Update, Delete) kapselt. Das Repository stellt eine einheitliche Schnittstelle für den Datenzugriff bereit und sorgt dafür, dass Änderungen an der Datenquelle (z. B. Firestore) keine Anpassungen in der restlichen Anwendung erfordern. Dadurch bleibt der Code übersichtlich, testbar und langfristig erweiterbar.

Die UML-Lern-App gliedert sich in fünf zentrale Funktionsmodule:

1. Benutzermanagement Registrierung, Login und Rollenprüfung
2. Kurs- und Unitverwaltung Laden und Anzeigen der Lerneinheiten
3. Quizsystem dynamische Anzeige von Fragen, Bewertung und Ergebnisdarstellung
4. Adminbereich Verwaltung von Kursen und Fragen
5. Notizenfunktion – persönliche Lernunterstützung durch eigene Einträge

Das grafische Design orientiert sich am Material Design von Google. Ziel war eine übersichtliche, barrierearme und responsive Oberfläche mit klarer Farbstruktur, moderner Typografie und konsistenten Interaktionselementen. Durch die Verwendung von ConstraintLayouts und Material Komponenten (z. B. Buttons, FloatingActionButtons und Cards) entstand ein modernes, benutzerfreundliches Erscheinungsbild, das auch auf unterschiedlichen Bildschirmgrößen stabil funktioniert.

## **4.3 Datenmodell und Firestore-Struktur**

Die Datenspeicherung der UML-Lern-App erfolgt über Firebase Cloud Firestore, eine dokumentenorientierte NoSQL-Datenbank. Firestore organisiert Daten in Collections (Sammlungen) und Documents (Dokumenten) und ist somit besonders flexibel und leistungsfähig für mobile Anwendungen. Dieses Modell ermöglicht eine hierarchische, leicht erweiterbare Datenhaltung, die speziell für Anwendungen mit wechselnder Internetverbindung geeignet ist. Die App greift in Echtzeit auf Firestore zu, wobei alle Daten – Benutzerprofile, Kurse, Fragen, Ergebnisse, Notizen und Feedbacks – zentral gespeichert und automatisch synchronisiert werden. Da Firestore Änderungen auch lokal zwischenspeichern kann, bleibt die App selbst bei einer unterbrochenen Internetverbindung voll funktionsfähig.

## 4.3.1 Zentrale Entitäten

Die Firestore-Datenbank bildet die Grundlage aller Anwendungsfunktionen. Im Mittelpunkt steht die Sammlung users, in der alle Benutzerprofile gespeichert werden. Jedes Profil enthält Informationen wie Benutzername, E-Mail-Adresse, Rolle (user oder admin), Punktestand und Lernfortschritt. Die Sammlung courses enthält sämtliche Lernkurse mit Titel, Beschreibung und Schwierigkeitsgrad. Innerhalb jedes Kurses sind die dazugehörigen units (Lerneinheiten) angelegt, die wiederum eine Reihe von questions (Fragen) enthalten. Diese Struktur spiegelt den Lernpfad der App wider: vom Kurs über die Lerneinheit bis zur einzelnen Frage. Weitere Sammlungen ergänzen den Funktionsumfang. Die Sammlung attempts dokumentiert die Quizversuche der Lernenden mit erzielter Punktzahl, Dauer und Abschlusszeit. Notes sind eine persönliche Subcollection innerhalb des Benutzer-Dokuments, die den Lernenden ermöglicht, eigene Anmerkungen oder Gedanken zu speichern. In der Sammlung feedback können Rückmeldungen und Bewertungen abgelegt werden, die von Administratoren ausgewertet werden können. Durch diese Organisation entsteht ein klar nachvollziehbares Datenmodell, das alle Kernprozesse der App unterstützt: Benutzerverwaltung, Kursorganisation, Quizsystem, Notizenverwaltung und Feedback-Funktion. Alle Entitäten sind als Kotlin-Data-Classes implementiert, wodurch eine direkte Verbindung zwischen Datenbankstruktur und Quellcode besteht.

## 4.3.2 Beziehungen zwischen den Entitäten

Die Beziehungen zwischen den Datenelementen sind logisch und übersichtlich aufgebaut. Ein Benutzer kann mehrere Quizversuche, Notizen und Feedback-Einträge besitzen, wodurch eine 1-n-Beziehung zwischen der Entität User und ihren abhängigen Daten entsteht. Ein Kurs besteht aus mehreren Einheiten, und jede Einheit wiederum aus mehreren Fragen – diese Beziehungen sind als Komposition umgesetzt, da untergeordnete Elemente ohne ihren übergeordneten Kontext nicht existieren. Das vollständige Klassendiagramm mit allen Entitäten, Beziehungen und Kardinalitäten befindet sich im Anhang A (Abbildung A.1). Es zeigt die genaue Struktur der Firestore-Sammlungen sowie die Zuordnung der Data Classes in Kotlin. Die dort dargestellten Beziehungen dienten als Grundlage für die Implementierung in Android Studio.

## 4.3.3 Sicherheit und Rollenmodell

Ein zentraler Bestandteil der Architektur ist das Sicherheits- und Rollenmodell, das den Zugriff auf Daten präzise steuert. Die UML-Lern-App nutzt Firebase Authentication zur Authentifizierung und Firestore Security Rules für den Zugriffsschutz. Jeder Benutzer wird eindeutig durch eine UID identifiziert und besitzt in der Sammlung users ein Feld role, das die jeweilige Rolle definiert. Es werden zwei Rollen unterschieden: User (Lernende) und Admin. User dürfen Quizze absolvieren, eigene Notizen erstellen und Feedback geben, haben jedoch keine administrativen Rechte. Admins dürfen zusätzlich Kurse und Fragen erstellen, bearbeiten und löschen. Beim Anmelden wird die Rolle automatisch aus Firestore geladen, damit die Oberfläche (z. B. Adminbereich) dynamisch angepasst werden kann. Dadurch ist gewährleistet, dass Lernende keine administrativen Bereiche betreten können.

Une image contenant texte, Police, ligne, nombre

Le contenu généré par l’IA peut être incorrect.

Diese Regeln stellen sicher, dass Benutzer ausschließlich ihre eigenen Daten lesen oder schreiben können und nur Administratoren Änderungen an Kursen oder Fragen vornehmen dürfen. Alle Verbindungen zwischen App und Firestore erfolgen verschlüsselt über HTTPS (TLS 1.2). Zusätzlich werden alle Benutzereingaben in der App validiert, um fehlerhafte oder unerlaubte Datenübertragungen zu verhindern. Damit ist ein robustes, rollenbasiertes Sicherheitssystem gewährleistet, das Datenschutz, Integrität und Stabilität sicherstellt.

## 4.3.4 Offline-Struktur und Synchronisation

Firestore unterstützt die lokale Datenspeicherung, wodurch alle Daten auch offline verfügbar sind. Wenn das Gerät keine Internetverbindung hat, werden Änderungen zunächst lokal zwischengespeichert. Sobald wieder eine Verbindung besteht, synchronisiert Firestore diese automatisch mit der Cloud. Dies betrifft insbesondere Notizen, Quizversuche und Feedback-Einträge, die Lernende jederzeit erfassen können. Die hierarchische Struktur folgt dabei festen Pfaden, z. B. /users/uid123/notes/note01 oder /courses/course01/units/unit01/questions/q01. Diese klare Organisation ermöglicht eine effiziente Abfrage, Synchronisation und spätere Auswertung der Daten. Sie trägt entscheidend zur Stabilität und Benutzerfreundlichkeit der Anwendung bei, insbesondere in schulischen Netzwerken mit eingeschränktem Internetzugang.

## 4.4 Benutzeroberfläche

Die grafische Gestaltung orientiert sich am Material Design von Google und folgt dem IAD-Farbschema mit einem kräftigen Rot (#C62828) als Akzentfarbe und Weiß als Hintergrund. Das Erscheinungsbild ist bewusst schlicht und klar gehalten, mit großzügigen Abständen, klarer Typografie und einem hohen Kontrast. Als Schriftart kommt Roboto zum Einsatz.Die wichtigsten Ansichten der App sind: die Login- und Registrierungsseite, die Kursübersicht, die Quiz-Ansicht, die Ergebnisanzeige, die Fehlerliste, die Notizverwaltung, die Feedback-Seite sowie die Einstellungsseite mit den Bereichen Allgemein, Konto, Datenschutz und Über die App. Nach der ersten Anmeldung wird zusätzlich ein kurzes Onboarding-Tutorial angezeigt, das die wichtigsten Funktionen vorstellt und danach deaktiviert wird.Für Administratorinnen und Administratoren existiert eine eigene Verwaltungsoberfläche, über die neue Kurse oder Fragen angelegt, geändert und gelöscht werden können. Alle Ansichten wurden zunächst als Mockups entworfen (siehe Anhang A5) und anschließend als XML-Layouts umgesetzt. Symbole stammen aus den „Outlined Material Icons“, und sämtliche Oberflächen sind responsiv.

## 4.5 Technischer Aufbau

Die App besteht aus mehreren Aktivitäten und Fragmenten, die zusammen die einzelnen Module bilden. Beim Start prüft eine Splash-Activity, ob der Benutzer bereits angemeldet ist. Wenn nicht, öffnet sich die Login-Activity; andernfalls erfolgt die Weiterleitung in die Hauptansicht. Die Main-Activity enthält eine Navigationsleiste, über die man zwischen Kursen, Fehlerliste, Notizen und Profil wechseln kann. Die Quiz-Funktion wurde besonders sorgfältig entworfen. Vor Beginn kann die gewünschte Anzahl an Fragen ausgewählt werden (z. B. 5, 10 oder alle). Die Fragen und Antwortoptionen erscheinen in zufälliger Reihenfolge, damit kein Lerneffekt durch Positionserkennung entsteht. Bei Multiple-Choice-Aufgaben werden Teilpunkte vergeben: richtige Häkchen erhöhen, falsche reduzieren die Punktzahl, jedoch nie unter null. Alle Aufgaben sind gleich gewichtet; jede trägt dieselbe Punktzahl. Zusätzlich stehen zwei Modi zur Verfügung: ein Übungsmodus ohne Zeitlimit und ein Prüfungsmodus mit Timer, bei dem die Bearbeitung automatisch beendet wird, sobald die Zeit abläuft. Nach Abschluss des Quiz wird das Ergebnis angezeigt. Dabei berechnet die App die erreichten Punkte und aktualisiert das Lernlevel. Falsch beantwortete Fragen werden automatisch in der Fehlerliste gespeichert, um sie später erneut üben zu können. Die App funktioniert auch offline, da Firestore Daten zwischenspeichert und Änderungen automatisch synchronisiert, sobald eine Internetverbindung besteht. Für die Verwaltung ist ein eigener Administrationsbereich vorgesehen, der nur für Konten mit Administratorrolle sichtbar ist. Änderungen an Kursen oder Fragen werden direkt in der Datenbank gespeichert. Siehe Aktivität diagramm im Anhang

# Implementierungsphase

Die Implementierungsphase umfasst die technische Realisierung der in der Entwurfsphase konzipierten Funktionen.

Schwerpunkt lag auf der Umsetzung der Benutzerverwaltung, der Quizlogik und der Firestore-Anbindung, um eine lauffähige Android-App zu schaffen.

Alle Module wurden modular aufgebaut, getestet und anschließend zu einer funktionierenden Gesamtanwendung zusammengeführt.

## 5.1 Implementierung der Datenstrukturen

Die Datenspeicherung erfolgt über Firebase Cloud Firestore, eine dokumentenorientierte NoSQL-Datenbank. Zu Beginn der Implementierungsphase wurden alle notwendigen Collections (Sammlungen) und Subcollections angelegt. Die wichtigsten Sammlungen sind users, courses, units, questions, attempts, notes und feedback. Jede dieser Sammlungen wurde durch eine entsprechende Kotlin-Datenklasse im Package data.model abgebildet. Beispiele sind User.kt mit Feldern für UID, Benutzername, E-Mail, Rolle, Punkte und Level, Course.kt mit Titel, Beschreibung und Levelanforderung sowie Question.kt mit Fragetext, Antwortmöglichkeiten und korrekter Lösung. Diese Struktur ermöglicht eine klare Trennung zwischen Anwendungslogik und Datenspeicherung. Über Repository-Klassen (z. B. UserRepository, QuizRepository) werden CRUD-Operationen gekapselt, damit ViewModels unabhängig von Firebase arbeiten können. Die Synchronisierung zwischen Firestore und App erfolgt über LiveData und Coroutines, wodurch Datenänderungen in Echtzeit übernommen werden. Zur Sicherung der Daten wurden Firestore Security Rules implementiert, die Zugriffe nach Benutzerrolle einschränken. Nur Admins dürfen Kurse und Fragen ändern, während Lernende ausschließlich ihre eigenen Dokumente lesen und schreiben dürfen. Die Datenstruktur wurde so gestaltet, dass Offline-Caching und automatische Synchronisierung unterstützt werden.

## 5.2 Implementierung der Benutzeroberfläche

Die Benutzeroberfläche wurde mit XML-Layouts in Android Studio entwickelt und folgt den Material-Design-Richtlinien von Google. Zur Gewährleistung einer einheitlichen Struktur kommen ConstraintLayout und LinearLayout zum Einsatz. Die Navigation zwischen den Hauptbereichen der App (Login, Kursübersicht, Quiz, Profil, Notizen) erfolgt über Activities und Buttons. Die Oberflächen wurden so entworfen, dass sie übersichtlich und barrierearm sind: hohe Kontraste, große Touch-Flächen und responsive Anordnung für verschiedene Displaygrößen. Das Farbschema orientiert sich am Corporate Design der IAD (Mischung aus Rot, Weiß und Grautönen). Zentrale Views sind activity\_login.xml, activity\_quiz.xml, activity\_profile.xml, activity\_admin.xml und activity\_notes.xml. Icons und Farben wurden mit der Android-Resource-API verwaltet. Die Einbindung von Animations-Effekten (z. B. Fade-In bei Activity-Wechseln) verbessert die Benutzererfahrung. Screenshots der fertigen Ansichten befinden sich im Anhang A.7.

## 5.3 Implementierung der Geschäftslogik

Die Geschäftslogik wurde gemäß dem MVVM-Muster umgesetzt. Die Verarbeitung von Benutzereingaben, Datenabrufen und Rückmeldungen geschieht über ViewModels mit Repository-Anbindung. Das LoginViewModel steuert Registrierung, Authentifizierung und Rollenprüfung über FirebaseAuth. Das QuizViewModel lädt die Fragen, prüft Antworten, berechnet den Punktestand und speichert das Ergebnis in attempts. Das NotesViewModel verwaltet CRUD-Operationen auf den Subcollections users/{uid}/notes. Die Anwendung nutzt Kotlin-Coroutines für asynchrone Operationen und stellt so eine flüssige Nutzererfahrung sicher. Die Fehlerbehandlung wird über try-catch-Blöcke und Firebase-Callbacks realisiert, wobei Fehlermeldungen als Toast-Benachrichtigungen angezeigt werden. Das gesamte System ist so konzipiert, dass es bei Ausfällen oder Offline-Phasen keine Datenverluste gibt. Lokale Änderungen werden im Cache gespeichert und sobald möglich synchronisiert. Für die Admins wurde eine separate Logik implementiert, die CRUD-Operationen in den Kurs- und Fragensammlungen erlaubt. Damit wurde die in Kapitel 4 beschriebene Architektur vollständig technisch umgesetzt. Die App ist nach Abschluss dieser Phase funktionsfähig, stabil und bereit für die Test- und Abnahmephase.

# 6 Testphase

Ziel der Testphase war die Überprüfung der Funktionsfähigkeit, Stabilität und Datensicherheit der UML-Lern-App. Alle Hauptmodule wurden mithilfe von **Black-Box-Tests** geprüft, bei denen ausschließlich des Verhaltens der Anwendung aus Benutzersicht bewertet wurde. Die Tests wurden von mehreren **Umschülerinnen und Umschülern der IAD-Marburg** sowie von der **Projektbetreuerin** durchgeführt. Getestet wurde auf verschiedenen Android-Geräten (Smartphones mit Android 10 bis 13) und im Emulator, um Kompatibilität und Leistung sicherzustellen. Im Mittelpunkt standen die Kernfunktionen der App: Die **Benutzerverwaltung** über Firebase Authentication ermöglichte eine fehlerfreie Registrierung, Anmeldung und Rollentrennung zwischen Lernenden und Administratoren.  
Mehrere Testpersonen legten neue Benutzer an und überprüften, ob Administratoren Kurse anlegen konnten, während Lernende nur auf Quiz- und Notizfunktionen zugreifen durften. Das **Quizmodul** zeigte Fragen korrekt an, berechnete den Punktestand und speicherte die Ergebnisse in Firestore. Bei einem Testlauf mit fünf Quizrunden wurden alle Resultate korrekt im Profil angezeigt, und der Timer beendete das Quiz zuverlässig nach Ablauf der Zeit. Das **Notizen-Modul** wurde von Lernenden genutzt, um eigene Einträge während des Quiz anzulegen. Ein Offline-Test im Flugmodus bestätigte, dass Notizen lokal zwischengespeichert und nach Wiederverbindung automatisch synchronisiert wurden. Auch das **Feedback-System** funktionierte fehlerfrei: Alle Rückmeldungen erschienen unmittelbar in der Firestore-Sammlung feedback, und das **Onboarding-Tutorial** wurde nur beim ersten App-Start eingeblendet. Die Anwendung zeigte in allen Tests eine hohe Stabilität, kurze Ladezeiten und eine fehlerfreie Synchronisierung mit Firebase. Überprüfungen der **Firestore Security Rules** bestätigten, dass nur autorisierte Benutzer auf ihre eigenen Daten zugreifen konnten. Kleinere Layoutanpassungen, etwa bei Schriftgrößen und Button-Abständen, wurden nach Rückmeldung der Testgruppe optimiert. Insgesamt verlief die Testphase erfolgreich: Sämtliche definierten Testfälle wurden positiv abgeschlossen, und die App erfüllt alle Anforderungen aus dem Projektantrag vollständig. Eine tabellarische Übersicht der einzelnen Testfälle befindet sich im **Anhang B1**.

# 7 Abnahmephase

Nach Abschluss der Testphase wurde die Anwendung der Projektbetreuerin vorgestellt und im Rahmen einer Demonstration auf einem Android-Gerät präsentiert. Ziel der Abnahme war die Überprüfung, ob alle im Projektantrag definierten Anforderungen vollständig umgesetzt wurden und die App stabil lauffähig ist. Während der Präsentation wurden die wichtigsten Funktionen vorgeführt: Benutzerregistrierung, Login, Kursauswahl, Quizdurchführung, Notizen, Feedback und die administrative Kursverwaltung. Alle Abläufe konnten ohne Fehlermeldung ausgeführt werden, und die Synchronisierung mit Firebase funktionierte in Echtzeit. Die Betreuerin bestätigte die Erfüllung der definierten Anforderungen. Besonders positiv hervorgehoben wurden die übersichtliche Benutzeroberfläche, das funktionierende Offline-Caching und die klare Trennung zwischen Lernenden und Administratoren. Kleinere Anpassungen an der Farbgestaltung und der Textgröße wurden anschließend vorgenommen, um die Lesbarkeit weiter zu verbessern. Damit gilt das Projekt als erfolgreich abgenommen. Die App erfüllt sowohl funktional als auch technisch alle Vorgaben und kann künftig als Grundlage für weitere Lerninhalte und Schulungseinheiten genutzt werden.

# Projektdokumentation und Fazit

Im Verlauf des Projekts wurden alle Schritte von der Analyse bis zur Abnahme sorgfältig + dokumentiert. Die technische Umsetzung wurde in Android Studio mit Kotlin und Firebase realisiert, wobei die Projektstruktur, Datenmodelle und Sicherheitsregeln in dieser Dokumentation vollständig beschrieben sind. Quellcode-Ausschnitte, Diagramme und Testergebnisse sind im Anhang aufgeführt. Rückblickend konnte das Projektziel – die Digitalisierung des Lernprozesses für UML-Diagramme – vollständig erreicht werden. Die App ermöglicht ein effizientes, motivierendes und ortsunabhängiges Lernen. Besonders erfolgreich war die Umsetzung der Offline-Funktion und der rollenbasierten Zugriffskontrolle, die das System sicher und flexibel macht. Für die Zukunft sind mehrere Erweiterungen geplant: Eine **Statistik-Funktion**, die Lernfortschritte grafisch darstellt, eine **Export-Option** für Ergebnisse (PDF/Excel) sowie ein optionales **Zertifikatsmodul** zur automatischen Bestätigung bestandener Einheiten. Langfristig könnte die App auch auf iOS portiert oder um neue Themenbereiche (z. B. SQL, Testing oder C#) ergänzt werden.

**Lessons Learned**

Während der Entwicklung der UML-Lern-App konnte wertvolle Praxiserfahrung im Umgang mit modernen Cloud-Technologien und der MVVM-Architektur gesammelt werden. Besonders deutlich wurde, wie wichtig eine saubere Strukturierung von Code und Datenmodellen für Wartbarkeit und Fehlersuche ist. Auch der Umgang mit Firebase – insbesondere die Einrichtung sicherer Zugriffsregeln und die Synchronisation zwischen Offline- und Online-Modus – erwies sich als lehrreich. Zudem zeigte sich, dass frühzeitiges Testen, Versionskontrolle über GitHub und eine klare Planung entscheidend für den Projekterfolg sind. Insgesamt führte das Projekt zu einem tieferen Verständnis für Softwarearchitektur, App-Design und agile Entwicklungsprozesse.

Die positive Resonanz und das stabile Endergebnis zeigen, dass die App nicht nur technisch funktioniert, sondern auch einen echten Mehrwert im Unterricht bietet. Damit stellt das Projekt einen wichtigen Meilenstein in meiner beruflichen Entwicklung dar und bildet die Grundlage für künftige, komplexere Softwareprojekte.