



**Westfälische
Hochschule**

CODEPILOT **EINE INTERAKTIVE LERNPLATTFORM FÜR JAVA IM RETRO-STIL**

Rezan Taylan / Dabir Ahmad Khan / Luca Wegener / Ricardo Szoltysek

IT-Basierte Innovationen SS 2025

Prof. Dr. Christian Kruse

Gruppe 6

Inhaltsverzeichnis

1. Einleitung	3
2. Seitenstruktur und Inhalte.....	4
3. Funktionale Umsetzung im Vergleich zum Lastenheft	6
4. Technische Umsetzung.....	7
5. Visuelle Projektgalerie	9
6. Lessons Learned	15
7. Fazit und Ausblick	16
Aufgabenverteilung	17

1. Einleitung

Im Rahmen des Moduls *IT-Innovationen SS 2025* haben wir das Projekt "**Code Pilot**" umgesetzt. Ziel war es, eine eigene Webanwendung zu entwickeln, die Studierenden beim Lernen von Java-Grundlagen hilft. Die Idee war, eine Plattform zu schaffen, die nicht nur Inhalte zeigt, sondern auf spielerische und motivierende Weise zum Lernen anregt. Dabei haben wir uns am offiziellen Lastenheft orientiert, in dem die Entwicklung eines AI-basierten Personal Teaching Assistant vorgesehen war.

Statt einer reinen KI-Integration haben wir uns für einen webbasierten Lerncoach entschieden, der einige zentrale Elemente aus dem Lastenheft umsetzt – zum Beispiel:

- interaktive Übungen (Multiple-Choice, Lückentext, Editor),
- ein Punktesystem mit XP, Level und Abzeichen,
- Fortschrittsanzeige und Feedback,
- visuelle Elemente wie Karten, Videos, Fun-Facts,
- eine persönliche Benutzerführung mit Avatar und Charakterauswahl.

Unser Fokus lag auf dem Thema *Grundlagen der Java-Programmierung*. Die Anwendung ist vollständig im Browser nutzbar und wurde mit HTML, CSS, JavaScript sowie einem Node.js-Backend umgesetzt. Die Inhalte sind modular aufgebaut und in verschiedenen Lerneinheiten organisiert.

In dieser Dokumentation zeigen wir den Aufbau unseres Projekts, erklären die Funktionen und stellen den Bezug zu den Anforderungen aus dem Lastenheft her.

2. Seitenstruktur und Inhalte

Unser Projekt besteht aus mehreren eigenständigen HTML-Seiten, die jeweils einen bestimmten Teil des Lernprozesses abdecken. Die Navigation erfolgt über eine einheitliche Retro-Navbar mit Benutzeranzeige, XP-Leiste und Avatar.

Start- und Loginbereich

- index.html: Begrüßungsseite mit Zugang zu Login/Registrierung.
- register.html: Registrierung mit Charakterauswahl (Karussell-Stil).
- login.html: Login-Maske mit einfachem Benutzerabgleich.
- seite0.html: Intro-Seite mit Willkommensnachricht.

Kursübersicht & Lerneinheiten

- seite1.html: Übersicht über den Kurs und den Fortschritt (XP, Übungen, Abzeichen).
- seite2.html: Einstieg in das Lernen – inklusive Java-Knowledge-Karten, Fun-Facts und Fortschrittsanzeige.

Wissensvermittlung

- wiki.html: Thematisch gegliederte Übersicht zu Java-Inhalten (Videos, Bibliothek, Platzhalter).
- Javabib.html: Horizontale Bibliothek mit scrollbaren Lerneinheiten als Karten.
- Javaws.html: Platzhalter

Interaktive Übungen

- quiz1.1.html bis quiz9.5.html: Über 45 Quizseiten mit Multiple-Choice- und Lückentextaufgaben.
- Jede Quizseite enthält: Fortschrittsanzeige, XP-Belohnung, Alertbox, Confetti-Effekt und einen Hinweis-Forscher.

Editor-Seite

- editor.html: Hier können Java-Codeübungen direkt im Browser ausprobiert werden (CodeMirror).
- Der Code wird mit einer Referenzlösung verglichen, richtige Lösungen geben XP und speichern den Fortschritt.

Profil & Fortschritt

- konto.html: Persönliche Statistik mit Level, Übungen, Badges, XP-Anzeige und Passwortänderung.

Technische Zusatzseiten

- ladebildschirm.html, test.html, loader.js: Dienen dem Ladeeffekt und der User Experience.

Alle Seiten sind durchgehend im Retro-Stil gehalten, mit Light-/Dark-Mode, eigens gestalteter Schrift und Benutzerführung.

3. Funktionale Umsetzung im Vergleich zum Lastenheft

Im Lastenheft wurden sechs zentrale funktionale Anforderungen (A–F) beschrieben, die wir mit unserem Projekt größtenteils abdecken konnten:

A: Nutzung vorhandener Kursmaterialien → Wir haben die wichtigsten Java-Grundlagen in einer eigenen Java-Bibliothek (Javabib.html) strukturiert aufbereitet. Diese besteht aus scrollbaren Karten mit folgenden Themen: 1) Hello World, 2) Variablen, 3) Kontrollfluss, 4) Schleifen, 5) Vererbung, 6) Arrays, 7) Methoden, 8) Klassen und Objekte, 9) Exception Handling. Zusätzlich wurden passende Videos, Fun-Facts, Theorie-Abschnitte und Übungsaufgaben ergänzt.

B: Individuelle Lernziele und Lernpfade → Zwar gibt es keine manuell einstellbaren Lernziele, jedoch kann der Nutzer frei entscheiden, in welcher Reihenfolge er die Quizthemen oder Theoriesektionen besucht. Die Fortschrittsanzeige zeigt, welche Inhalte bereits bearbeitet wurden.

C: Adaptive Anpassung des Curriculums

→ Tatsächlich haben wir mit der index.html auch eine einfache KI-Integration umgesetzt. Dort können Nutzer dem eingebauten KI-Chat (basierend auf einem LLM) Fragen stellen – zum Beispiel zu Java-Themen, Codeproblemen oder allgemeinen Begriffen. Zusätzlich speichern wir den Chatverlauf lokal, sodass man jederzeit nachsehen kann, welche Fragen man bereits gestellt hat oder welche Themen behandelt wurden. Damit wird zwar kein automatischer Lernpfad generiert, aber eine adaptive und nachvollziehbare Unterstützung ermöglicht – ganz im Sinne eines personalisierten Lernassistenten.

D: Interaktive Problemlösung → Vollständig umgesetzt durch Lückentexte, Multiple-Choice-Fragen und Codeeditor-Aufgaben. Jede Aufgabe prüft Wissen schrittweise und gibt bei Erfolg visuelles Feedback (XP, Konfetti).

E: Multimodale Lehrinhalte → Erfüllt: Es gibt Textkarten, Bilder, eingebettete YouTube-Videos, Animationen (z. B. Forscher mit Sprechblase) und interaktiven Java-Code.

F: Personalisiertes Feedback mit Gamification → Vollständig umgesetzt mit XP, Level, Fortschrittsbalken, Abzeichen und Charakterauswahl. In der Navbar wird der Fortschritt ständig angezeigt.

Insgesamt orientiert sich unser Projekt stark an den Vorgaben des Lastenhefts, wobei der Fokus klar auf der Umsetzung eines motivierenden, eigenständigen Java-Lernsystems liegt – mit vielen Features, die Studierenden das Lernen erleichtern und unterhaltsamer machen.

4. Technische Umsetzung

Unser Projekt wurde vollständig mit Webtechnologien umgesetzt. Das Frontend besteht aus HTML, CSS und JavaScript, wobei wir bewusst auf Frameworks wie React oder Vue verzichtet haben, um die Funktionsweise möglichst transparent und nachvollziehbar zu halten.

Frontend

- **HTML & CSS:** Das Grundgerüst jeder Seite besteht aus statischem HTML mit einem einheitlichen Layout. Für das Styling wurde klassisches CSS genutzt – ergänzt durch mehrere Stylesheets für Retro-Design, Light-/Dark-Mode und Pixel-Effekte.
- **JavaScript:** Für die interaktive Logik wurde JavaScript verwendet, unter anderem für:
 - Dark-/Light-Mode Toggle
 - Fortschrittsberechnung & LocalStorage-Verwaltung
 - Konfetti-Effekte und Alertboxen
 - Dynamisches Nachladen von Quizinhalten und Validierung von Antworten
 - Avatar-Auswahl und Chatsteuerung (inkl. Speicherung der Chatverläufe)
- **CodeMirror:** Auf der editor.html-Seite kommt CodeMirror zum Einsatz, um Java-Code direkt im Browser editierbar zu machen. Hier kann der Nutzer einfache Übungen wie „Hello World“ oder Schleifen schreiben und automatisch überprüfen lassen.

Backend

- **Node.js mit Express:** Das Backend basiert auf Node.js und Express. Es stellt verschiedene REST-API-Endpunkte zur Verfügung, etwa für:
 - Benutzer-Login und -Registrierung
 - Passwortänderung
 - Fortschritts-Update (XP, Übungen)
 - Abzeichenverwaltung (freischaltbar alle 5 Übungen)
- **JSON-Dateien:** Nutzer- und Fortschrittsdaten werden in einer users.json-Datei gespeichert und ausgelesen. Für einfache Prototypen-Tests wurde bewusst auf eine relationale Datenbank verzichtet.

Datenfluss & Speicherung

- **Frontendseitige Speicherung:** Fortschritt (XP, erledigte Übungen, Chatverlauf) wird lokal im localStorage bzw. sessionStorage gespeichert.
- **Serverseitige Speicherung:** Bei erfolgreichem Abschluss von Übungen werden die aktualisierten Werte per API an den Server gesendet und in der JSON-Datei gespeichert.

API-Anbindung der KI-Chatfunktion

- **KI-Chat auf index.html:** Die Chatfunktion basiert auf einem einfachen lokalen LLM-Frontend, das über eine API mit einem externen Sprachmodell kommunizieren kann. Dabei wird eine POST-Anfrage mit der Nutzerfrage an einen definierten API-Endpunkt gesendet. Die Antwort der KI wird direkt im Frontend angezeigt und gleichzeitig lokal im Chatverlauf gespeichert. So entsteht eine personalisierte Hilfestellung, die individuell auf Nutzerfragen reagiert.

Weitere technische Details

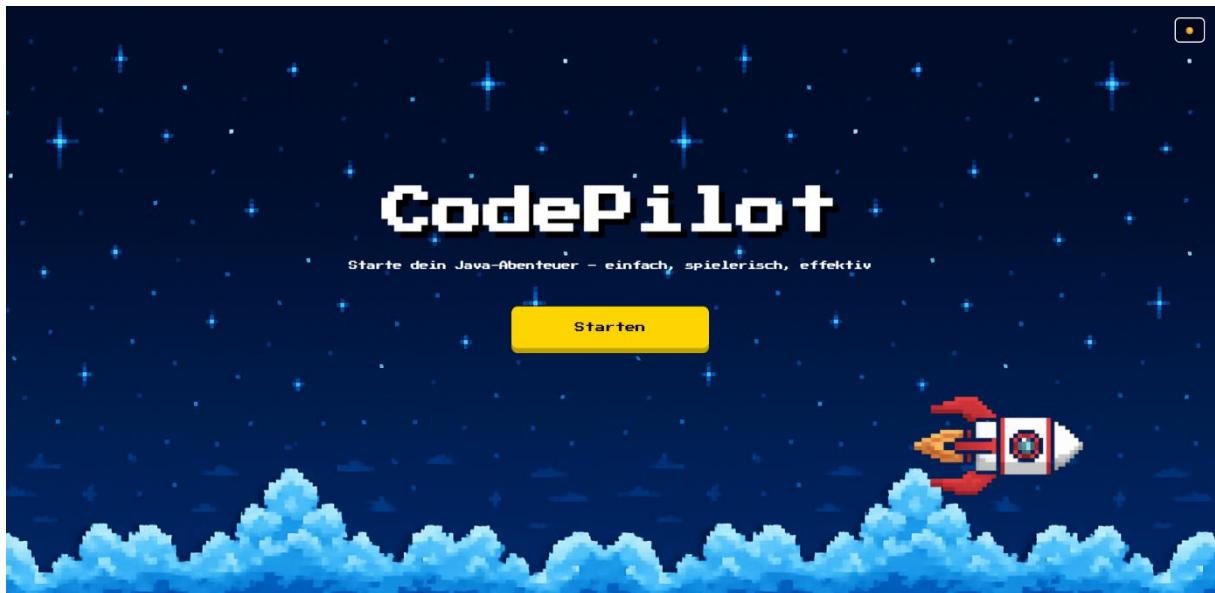
- **Chat-Funktion (LLM):** Der Chat auf index.html nutzt eine einfache lokale Verarbeitung (optional mit Integration eines KI-Dienstes). Fragen werden gespeichert, sodass ein Verlauf entsteht.
- **Responsive Design:** Die Seiten sind für Desktop optimiert, funktionieren aber auch auf Tablets. Auf Bootstrap wurde bewusst verzichtet.
- **Ladebildschirm:** Vor Quizstart wird ein animierter Ladebildschirm eingeblendet (test.html), der den Nutzer zum jeweiligen Ziel weiterleitet.

Diese einfache, aber effektive technische Architektur erlaubt es uns, das System flexibel zu erweitern, z. B. mit neuen Quizformaten, einem erweiterten Backend oder einer Datenbankanbindung.

5. Visuelle Projektgalerie

1. Startseite

Die Startseite bildet den Einstieg in die Lernplattform *CodePilot* und lädt Nutzer mit einem ansprechenden Pixel-Design zur Interaktion ein. Mit einem einfachen Klick auf „Starten“ beginnt das Java-Abenteuer – spielerisch, motivierend und klar strukturiert.



2. Login-Seite

Die Login-Seite bildet den Einstiegspunkt in die Anwendung. Nutzer geben hier ihre Zugangsdaten ein, um auf den geschützten Bereich der Lernplattform zuzugreifen. Das Design greift den Retro-Stil des Projekts auf und kombiniert eine klare Benutzerführung mit spielerischer Atmosphäre.



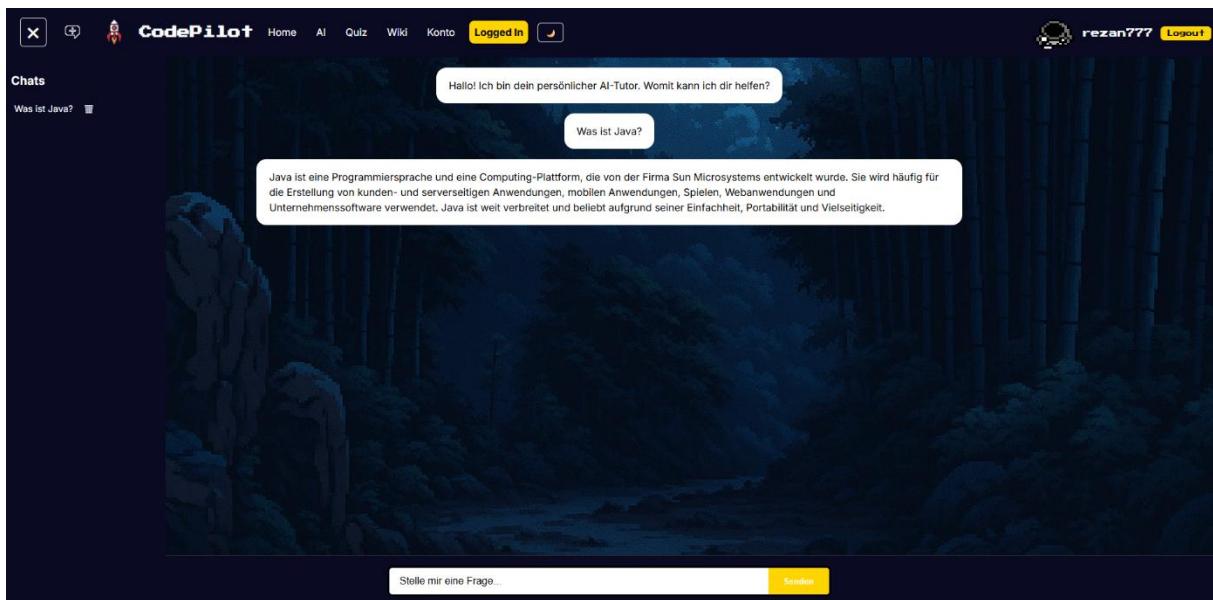
3. Registrierungsseite

Auf der Registrierungsseite können neue Nutzer ein Konto erstellen und dabei einen individuellen Charakter auswählen. Eine Kurzbeschreibung stellt jede Figur humorvoll vor, was den Gamification-Ansatz des Projekts unterstützt und zur Nutzerbindung beiträgt.



4. Chat mit KI-Tutor

Die integrierte Chat-Seite ermöglicht dem Nutzer, Fragen zur Programmiersprache Java an einen persönlichen KI-Tutor zu stellen. Die Antworten erscheinen im typischen Chat-Stil und bieten eine interaktive Möglichkeit, Lerninhalte eigenständig zu vertiefen.



5. Kursübersicht & Fortschrittsanzeige

Die Kursübersicht zeigt alle Kapitel und Übungen des Java-Kurses, gegliedert nach Themenbereichen. Auf der rechten Seite wird der persönliche Fortschritt visualisiert: erledigte Übungen, XP, Abzeichen und Streaks motivieren zur kontinuierlichen Arbeit.

The screenshot shows the CodePilot platform's course overview. On the left, a sidebar lists chapters: 1 Hello World, 2 Variablen, 3 Kontrollfluss, 4 Schleifen, 5 Vererbung, 6 Arrays, and 7 Methoden. Chapter 1 is expanded, showing five exercises: Übung 1 (Konsole aktiviert!), Übung 2 (Der Fall Java vs. JavaScript), Übung 3 (// Ready to Comment), Übung 4 (Mission: Hello World()), and Übung 5 (Level-Up: Erste Ausgabe). Each exercise has a blue "Start" button. On the right, there's a summary section titled "Dein Fortschritt" (Your Progress) with metrics: ÜBUNGEN (2), GESAMT-XP (40), Kursabzeichen (0), and TÄGLICHE STREIFEN (0). Below that is a section for "Kapitel-Badges" (Chapter Badges) featuring various icons. At the bottom, navigation buttons include Hinweis (Hint), Zurück (Back), Erledigt (Mark as done), and Nächste (Next).

6. Multiple-Choice-Aufgabe

In dieser interaktiven Quizseite lösen Nutzer eine klassische Multiple-Choice-Aufgabe auf Grundlage eines Java-Codes. Die Frage und Antwortoptionen sind im Retro-Stil gestaltet und ermöglichen eine direkte Rückmeldung bei Auswahl.

The screenshot shows a multiple-choice question in the quiz section. The question asks: "Was ist die richtige Ausgabe dieses Programms?" (What is the correct output of this program?). It displays a Java code snippet:

```
public class Hello { public static void main(String[] args) { System.out.println("Java"); } }
```

. To the right, a list of four answer options is shown in a retro-style interface: A) Hello World!, B) Java!, C) java, and D) Syntax Error. The first option, A, is selected with a radio button. Navigation buttons at the bottom include Hinweis (Hint), Zurück (Back), Erledigt (Mark as done), and Nächste (Next).

7. Lückentext-Aufgabe

Diese Übung kombiniert Java-Code und Theorie in Form einer Lückentext-Aufgabe mit Dropdown-Menüs. Nutzer müssen fehlende Begriffe korrekt einsetzen, um Syntax, Struktur und Begriffe wie main oder System.out.println zu verstehen.

The screenshot shows the CodePilot interface. On the left, a code editor window titled "Mission: Hello World()" displays a Java program:public class HelloWorld { public static void main(String[] args) { System.out.println("Hello, CodePilot!"); }}Specific words in the code are highlighted with dropdown menus, such as "HelloWorld" and "main". On the right, a separate window titled "Erklärung - Hello World" contains explanatory text:

Ein Java-Programm beginnt mit der Definition einer Klasse. Der Dateiname muss mit dem -- auswählen -- übereinstimmen.

Der Einstiegspunkt ist die -- auswählen --, die statisch ist und keinen Rückgabewert hat.

Als Parameter bekommt sie ein -- auswählen -- übergeben.

Die Ausgabe erfolgt mit -- auswählen --, wobei der Text in -- auswählen -- steht.

8. Java-Editor mit Konsole

Der integrierte Java-Editor erlaubt es den Nutzern, eigenen Code zu schreiben und direkt auszuführen. Bei korrekter Ausgabe wird der Fortschritt gespeichert. Dieses Format fördert aktives Lernen durch praktisches Ausprobieren.

The screenshot shows the CodePilot interface. On the left, a code editor window titled "Level-Up: Erste Ausgabe" displays a Java program:public class HelloWorld { public static void main(String[] args) { System.out.println("Hello, CodePilot!"); }}A button labeled "Run" is visible. On the right, a terminal window titled "HelloWorld.java" shows the output:Hello, CodePilot!

9. Java-Wiki-Übersicht

Die Wiki-Seite bietet eine visuelle Themenübersicht zu zentralen Java-Konzepten wie Variablen, Schleifen oder Vererbung. Jede Karte führt zu einer eigenen Lerneinheit mit Erklärungen und Beispielen. Das Design erinnert an eine digitale Bibliothek im Retro-Stil.



10. Java-Wiki-Beiträge

Ein Beispielbeitrag aus der Wiki-Sektion zum Thema „Exception Handling“. Er erklärt verschiedene Fehlerarten in Java und zeigt anhand von Tabellen und Codebeispielen, wie Ausnahmen mit try-catch behandelt werden können. Fachlich fundiert und strukturiert aufbereitet.

A screenshot of a Java-Wiki article titled "Exception Handling". The page has a dark background with bookshelves on either side. At the top, there's a navigation bar with links for Home, AI, Quiz, Wiki, Konto, and Logout. Below the navigation is a search bar labeled "Zuletzt aufgerufen". The main content area starts with a section titled "Arten von Fehlern" (Types of Errors) with the sub-section "In Java unterscheidet man drei Hauptarten:" (In Java, three main types are distinguished:). It includes a table:

11. Konto-Übersicht

Die Konto-Seite gibt den Nutzenden einen Überblick über ihren Lernfortschritt: erreichte XP, absolvierte Übungen, gesammelte Badges und zuletzt angesehene Inhalte. Ein individualisierbares Banner sowie ein persönlicher Avatar runden die Seite ab.

The screenshot shows the CodePilot account overview page. At the top, there's a banner with a landscape illustration and a user profile placeholder for 'rezan777'. Below the banner, the page is divided into several sections:

- Profil:** Shows the user level (Level 2), a quote ("Nullbit - Ein gescheitertes Experiment der 8-Bit-Wissenschaft. Kein Gesicht, kein Name - nur Code."), and a section for "Kapitel-Badges".
- Zuletzt angesehen:** A grid of three cards showing recent activity: "Zeit in der Wiki", "Erfolgreiche Übungen", and "Zeit auf der Videothek".
- Statistiken:** Summary statistics: ÜBUNGEN 3, GESAMT-XP 60, Kursabzeichen 0, and TUTORIALS 0.

12. Videothek

Die Videothek präsentiert eingebettete YouTube-Lernvideos sortiert nach Themen. Nutzer können Java-Einheiten wie „Hello World“ oder „Arrays“ direkt im Browser ansehen. Die Seitennavigation ermöglicht einen schnellen Zugriff auf relevante Inhalte.

The screenshot shows the CodePilot video library page. On the left, a sidebar lists video categories: Themen, Hello World, Hallo Welt, Variablen, Kontrollfluss, Schleifen, Arrays, Methoden, Klassen & Objekte, Vererbung, and Exception Handling. The main area displays a video player for a Java tutorial titled "#1 Java Tutorial Deutsch (German) [1/24] - Hallo Welt". The video player interface includes a play button, a progress bar showing 0:00 / 0:00, and a like/dislike counter. The background features a stone wall with torches.

6. Lessons Learned

Während der Umsetzung des Projekts „Code Pilot“ haben wir viele praktische Erfahrungen gesammelt – sowohl technisch als auch im Hinblick auf Nutzerfreundlichkeit.

Was gut funktioniert hat:

- Die klare Struktur der Seiten und die Aufteilung in Lern-, Quiz- und Profilbereiche hat sich als sehr hilfreich erwiesen.
- Die Integration von XP, Level und Badges hat die Motivation der Tester:innen gesteigert.
- Die Umsetzung im Retro-Stil mit Dark-/Light-Mode wurde von vielen positiv aufgenommen.
- Die KI-Chatfunktion hat sich als hilfreiche Ergänzung zur Selbsthilfe bei Verständnisfragen gezeigt.

Herausforderungen:

- Die JSON-basierte Benutzerverwaltung ist zwar einfach umzusetzen, aber langfristig nicht ideal für viele Nutzer.
- Das Styling der vielen Einzeldateien war aufwendig und musste manuell auf allen Seiten synchron gehalten werden.
- Die Kombination aus Code-Editor, automatischer Bewertung und XP-Tracking war technisch anspruchsvoll – insbesondere beim Vergleich von Ausgaben und beim Handling kleiner Syntaxfehler.

Lerneffekte:

- Wir haben viel über client- und serverseitige Speicherung gelernt (LocalStorage vs. JSON-API).
- Das Entwickeln einer durchgängigen Benutzerführung mit Gamification war herausfordernd, aber lohnenswert.
- Der Einsatz eines KI-basierten Helfers im Lernprozess erfordert klare Fragestellungen und eine gute Integration in das Gesamtsystem.

Insgesamt konnten wir durch dieses Projekt nicht nur unser technisches Wissen ausbauen, sondern auch erleben, wie wichtig eine gute Benutzerführung und motivierende Elemente beim Lernen sein können.

7. Fazit und Ausblick

Mit "Code Pilot" haben wir eine funktionale, motivierende und thematisch klar strukturierte Lernplattform für Java-Grundlagen entwickelt. Die Kombination aus interaktiven Übungen, Gamification und personalisierter Lernführung hat sich als effektiv erwiesen, um Nutzer:innen beim eigenständigen Lernen zu unterstützen.

Trotz einiger technischer Herausforderungen (z. B. Synchronisation der Inhalte, JSON-Handling) konnten wir ein stabiles und nutzbares System realisieren, das bereits in einer Testumgebung funktioniert.

Für die Zukunft wären folgende Erweiterungen denkbar:

- Anbindung an eine relationale Datenbank (z. B. SQLite oder MongoDB) zur besseren Verwaltung mehrerer Nutzer:innen.
- Erweiterung der KI-Integration, z. B. durch automatische Quizgenerierung oder inhaltliches Feedback.
- Mehrsprachigkeit (Deutsch/Englisch) zur breiteren Nutzbarkeit.
- Erweiterung des Kursinhalts über Java hinaus (z. B. Python, Webentwicklung).

Wir sind mit dem Ergebnis zufrieden und sehen „Code Pilot“ als gute Grundlage für weitere Studien- oder Praxisprojekte im Bereich E-Learning und Webentwicklung.

Aufgabenverteilung

Aufgabe

Verantwortliche Person(en)

Projektidee & Konzeption	Rezan, Dabir
Struktur des Lastenhefts prüfen	Rezan, Dabir
Design der Webseite (Layout & Stil)	Rezan, Dabir
Umsetzung Login-/Registrierungssystem	Rezan, Dabir
Navigation, Navbar & Benutzeranzeige	Rezan, Dabir
Entwicklung der Multiple-Choice-Quizseiten	Rezan, Dabir
Erstellung der Lückentext-Aufgaben	Rezan, Dabir, Ricardo, Luca
Java-Editor-Aufgaben mit Bewertung	Rezan, Dabir
Backend (Node.js, Express, API-Anbindung)	Rezan, Dabir
Fortschritts- & XP-System	Rezan, Dabir
Badges & Levelsystem	Rezan, Dabir
Java-Bibliothek mit Beiträgen im Wiki-Stil (Javabib.html)	Rezan, Dabir, Ricardo, Luca
Java-Videothek mit Sidebar (videos.html)	Rezan, Dabir
Integration des KI-Chats (index.html, API, Verlauf)	Rezan, Dabir
Speicherung & Synchronisation (LocalStorage + JSON-API)	Rezan, Dabir
Charakter-Auswahl mit Karussell & animierter Vorschau	Rezan, Dabir
Implementierung des Pixel-Forschers mit Hinweis-Sprechblase	Rezan, Dabir
Zufällige Tipps für Java-Themen (z. B. in der Kursübersicht)	Rezan, Dabir
Animation der Sprechblasen (Typing-Effekt, Timing, Sound)	Rezan, Dabir
Gestaltung der Fun-Fact-Karten im Wiki/Kursbereich	Rezan, Dabir

Aufgabenverteilung

Aufgabe

Verantwortliche Person(en)

Umsetzung des Ladebildschirms mit Weiterleitung (test.html)	Rezan, Dabir
Konfetti-Effekt bei richtiger Antwort	Rezan, Dabir
Eigene Alertbox im Retro-Stil (statt alert())	Rezan, Dabir
Dark-/Light-Mode-Umschaltung mit Speicherung	Rezan, Dabir
Fortschrittsbalken & visuelle XP-Anzeige	Rezan, Dabir
Badge-Freischaltung & grafische Darstellung	Rezan, Dabir
Dynamische Anzeige des aktuellen Charakters auf allen Seiten	Rezan, Dabir
Responsive Design für Quiz- und Kursseiten	Rezan, Dabir
Struktur & Gruppierung der Sidebar-Themen in der Videothek	Rezan, Dabir
Erstellung von YouTube-Einbettungen mit passenden Kapiteltiteln	Rezan, Dabir
Tests, Debugging & Fehlerbehebung	Rezan, Dabir
Dokumentation & Präsentation	Dabir, Rezan
Learner Journey	Luka, Ricardo