



Abschlussprüfung „Angular“ NE4NE8

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur Kurs-Projektarbeit: Earth Oceans Learning App

Entwicklung einer kindgerechten Angular SPA zum Rätsel rund um die sieben Meere. Jeder Schritt bringt dich näher zum Meeresprofi!

Ozeane für Kinder (7–9 Jahre)

Abgabe der Dokumentation: 09.12.2025

Prüfling: **Adil Sathi**

Further Str. 3-5

41462 Neuss

Ausbildungsbetrieb:



Inhaltsverzeichnis

1. Einleitung	5
2. Projektplanung	6
3. Analyse & Entwurf	10
4. Realisierung	16
5. Qualitätssicherung	22
6. Wirtschaftlichkeitsbetrachtung	26
7. Fazit & Ausblick	29
A. Anhang	33

1. Einleitung

1.1 Ausgangssituation

Während meiner Umschulung zum Fachinformatiker für Anwendungsentwicklung habe ich als Abschlussprojekt die „Earth Oceans Learning App“ entwickelt. Ziel war die Erstellung einer kindgerechten Angular Single Page Application (SPA), die Wissen über die sieben Ozeane vermittelt. Die App lädt Kinder dazu ein, fünf Kontinente zu bereisen, spannende Meerestiere zu entdecken und abwechslungsreiche Rätsel zu lösen – so werden sie Schritt für Schritt zum Meeresprofi.

1.2 Projektidee und Zielsetzung

Die App soll Kindern spielerisch Wissen über die sieben Ozeane vermitteln. Kernfunktionen:

- **Entdecke Meere und Tiere:** Detailseiten mit Informationen und Bildern.
- **Quiz:** Multiple-Choice-Fragen zur Wissensüberprüfung.
- **Puzzle:** Spielerisches Lernen durch Puzzle-Elemente.

1.3 Projektbegründung

Digitale Lernlösungen sind für Kinder motivierender als statische Materialien. Die App bietet Interaktivität, Barrierefreiheit und kann offline genutzt werden.

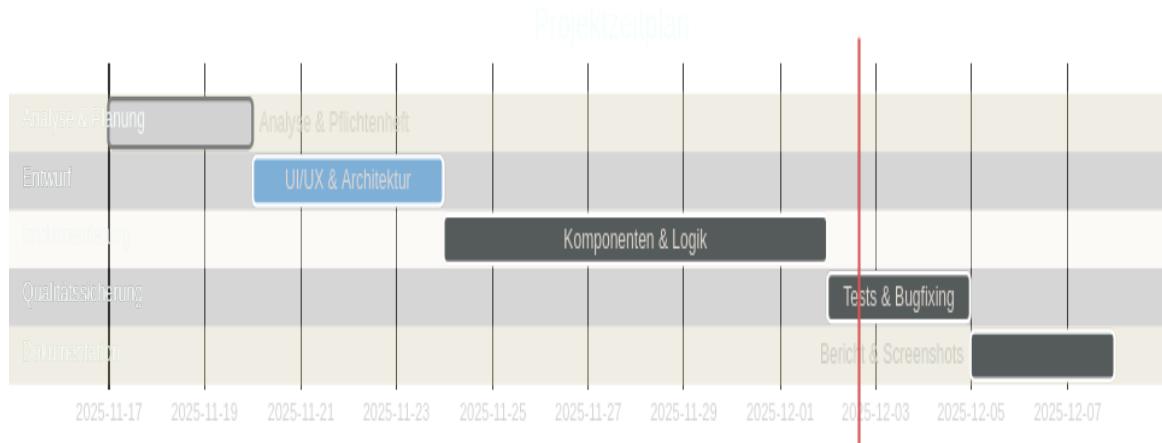
1.4 Make-or-Buy Entscheidung

Da gängige Lösungen wie digitale Puzzles oder Quiz-Anwendungen die geforderte Offline-Funktionalität und den gewünschten Datenschutz nicht sicherstellen, entschied ich mich für eine maßgeschneiderte Eigenentwicklung.

2. Projektplanung

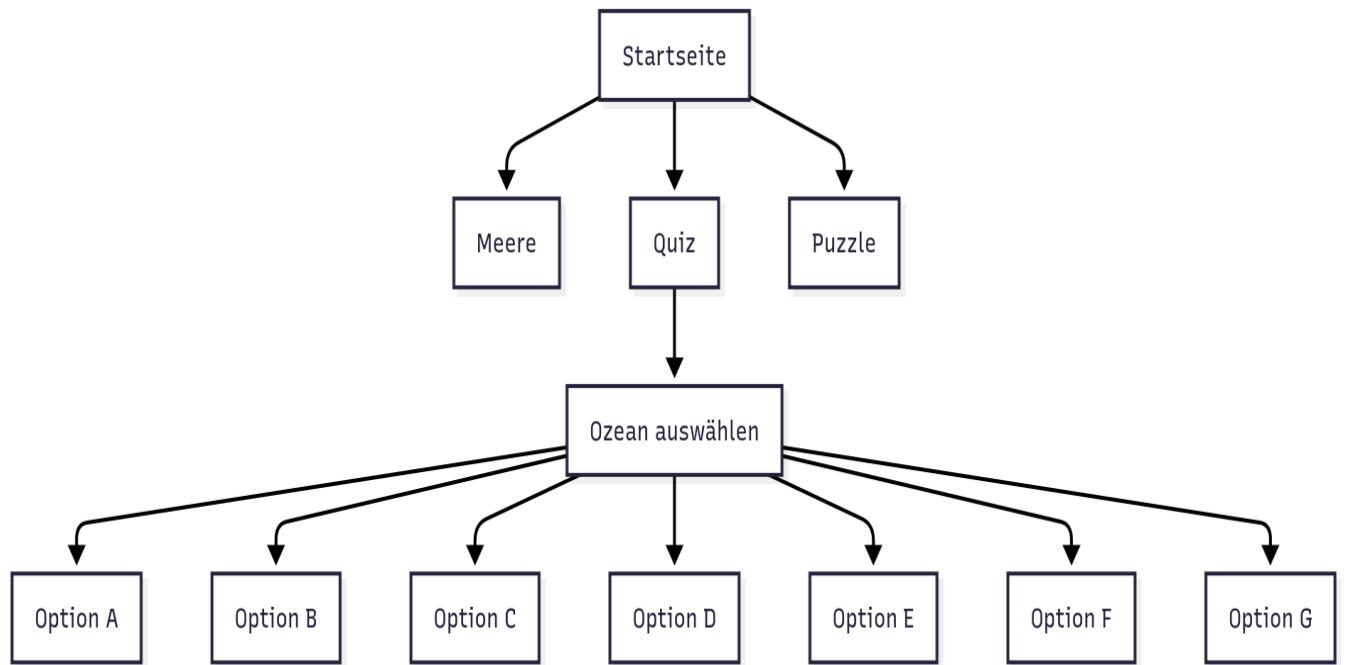
Zeitplanung (Tabelle):

Phase	Tätigkeit	Geplante Zeit (h)
Analyse & Planung	Ist-/Soll-Analyse, FCPP	27
Entwurf	UI/UX, Architektur	36
Implementierung	Komponenten, Logik	81
Qualitätssicherung	Tests, Bugfixing	27
Dokumentation	Bericht, Screenshots	45
Gesamt		216 h



3. Analyse & Entwurf

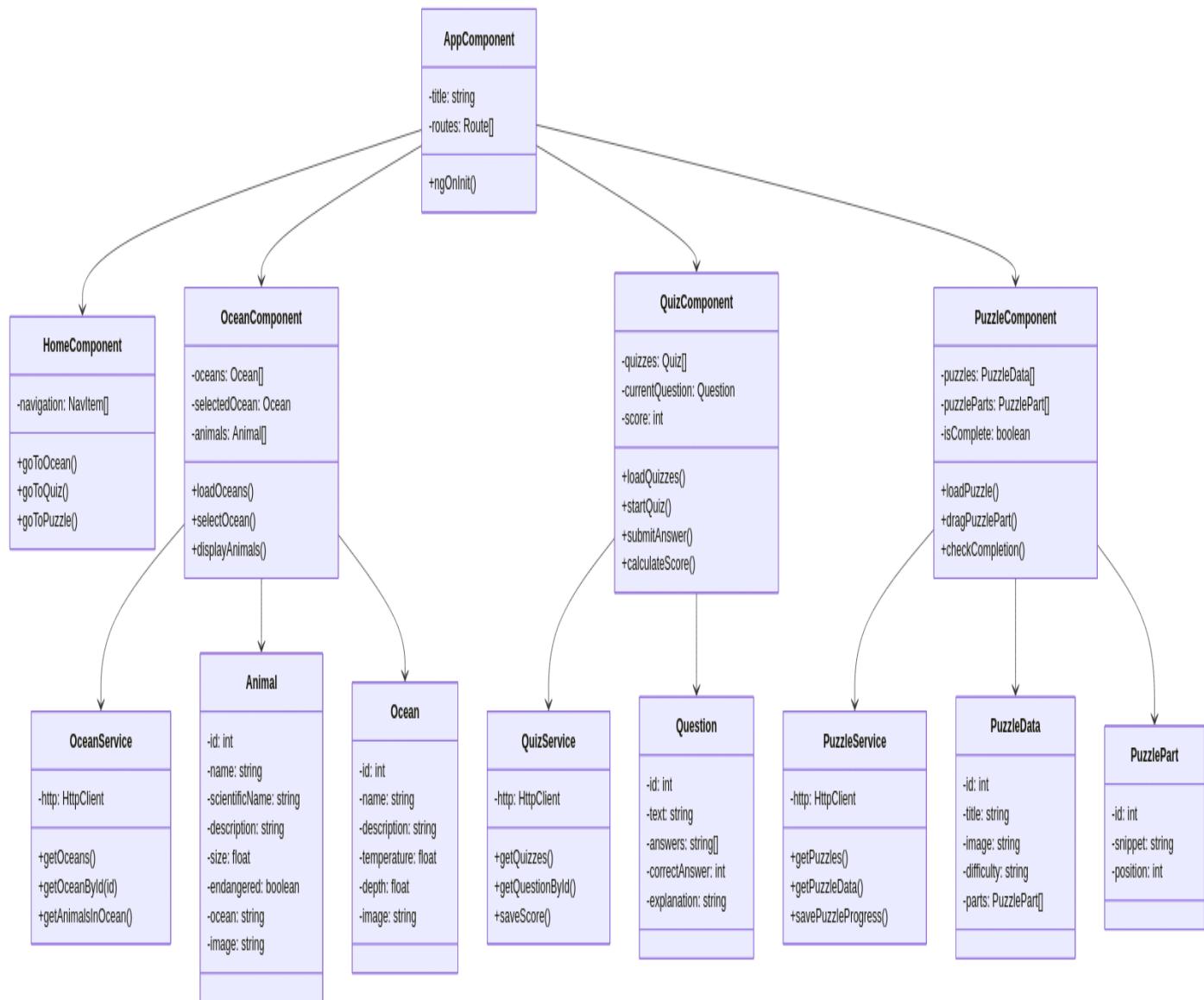
Use Case Diagramm (Mermaid):



Architekturdiagramm (Mermaid):



Klassendiagramm (Mermaid):



4. Realisierung

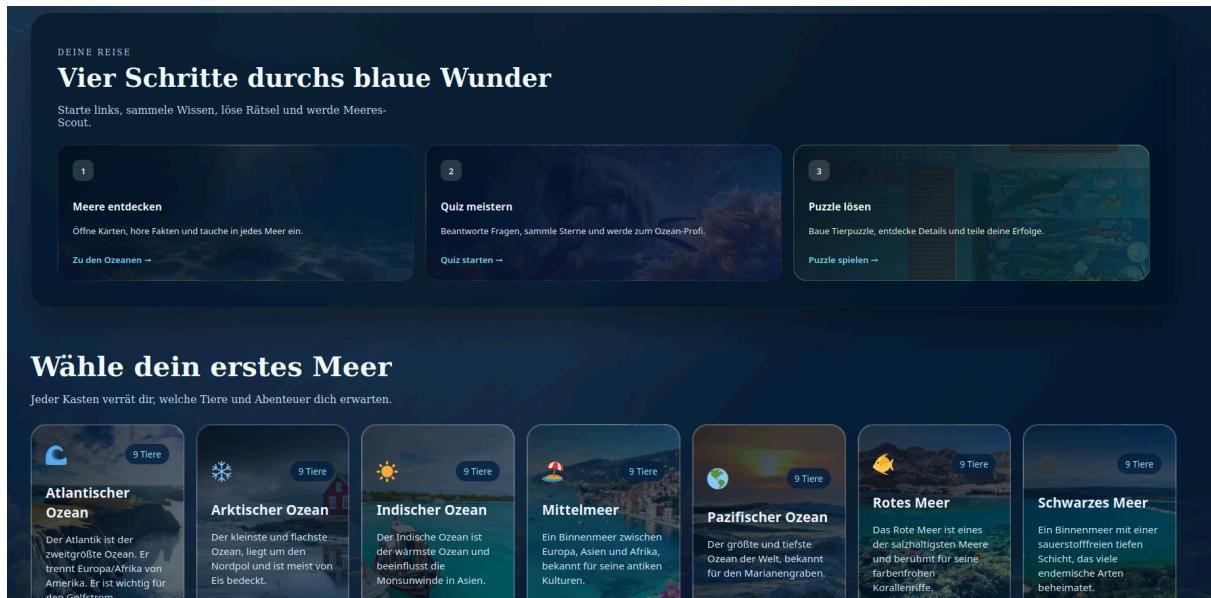
Listing 4.1: Beispiel Angular-Komponente

```

1  @Component({
2      selector: 'app-ocean-selection',
3      templateUrl: './ocean-selection.component.html',
4      styleUrls: ['./ocean-selection.component.css']
5  })
6  export class OceanComponent {
7      oceans = ['Atlantik', 'Pazifik', 'Indischer Ozean', 'Arktis',
8      'Rotes Meer', 'Swartz Meer', 'Mittelmeer']; }
  
```

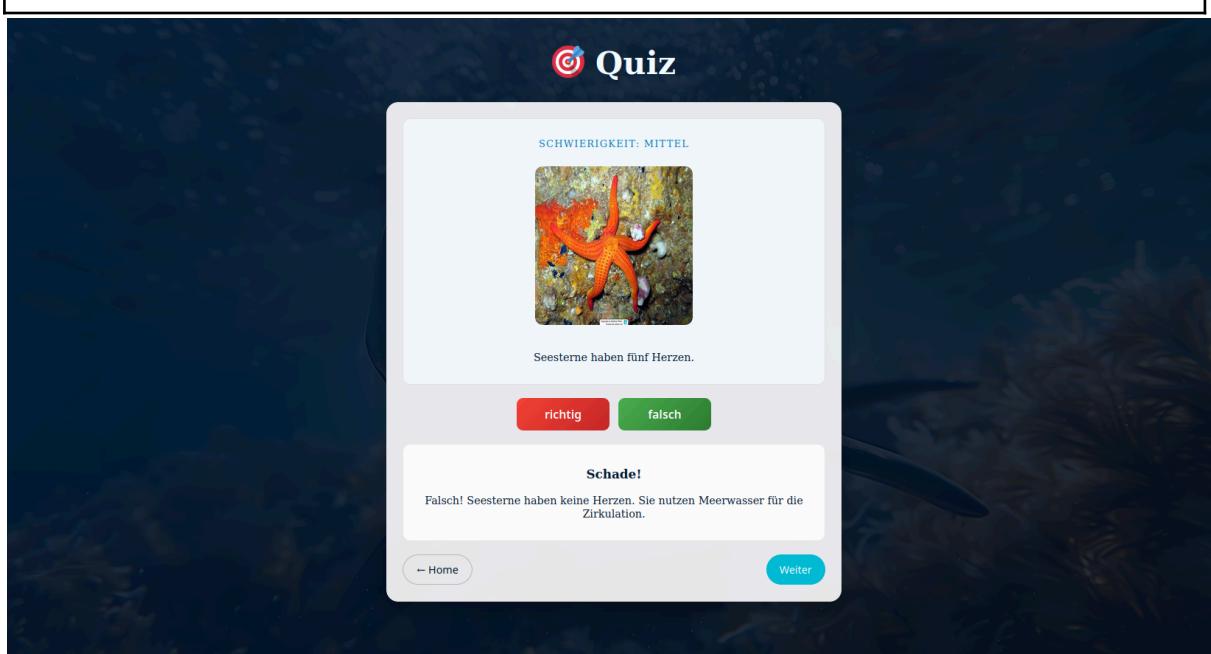
Screenshots der App (Startseite, Ozean-Detail, Quiz, Puzzle)

Listing 4.1: Home



Listing 4.2: Quiz

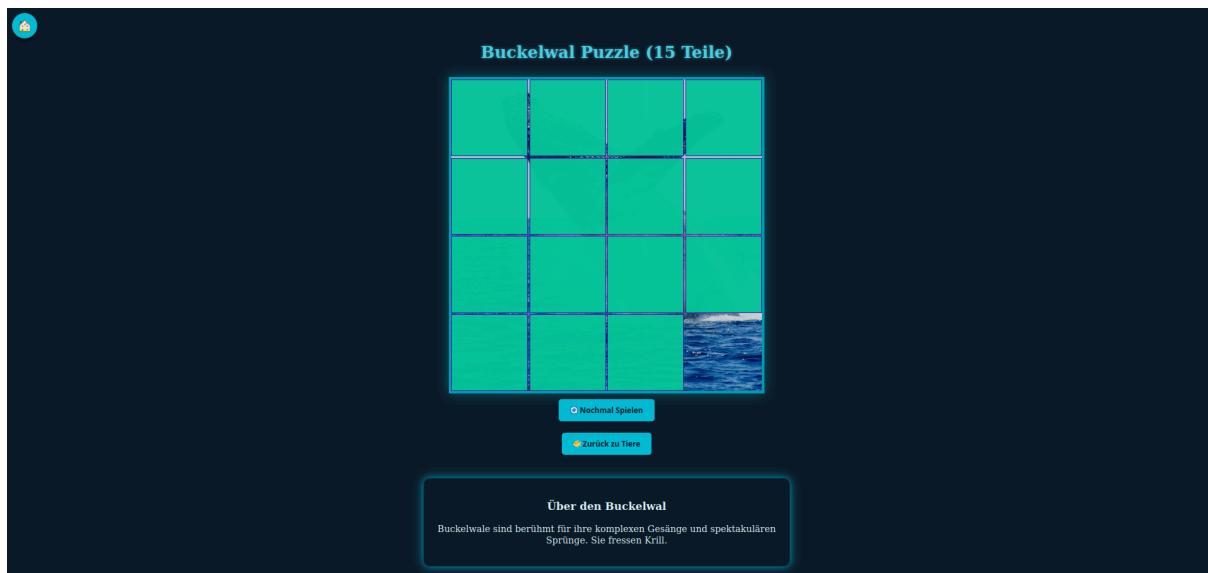
```
1  @Injectable({ providedIn: 'root' })
2  export class Quiz {
```



Listing 4.3: Ozean und Tiere



Listing 4.4: Puzzle



5. Qualitätssicherung

5.1 Chronologisches Projekt-Journal

Phase 1: Visuelles Design und Struktur-Updates

Die Startseite (Homepage) wurde umfassend neu gestaltet, um die Benutzerfreundlichkeit und Ästhetik zu verbessern. Dabei wurden die drei Hauptfunktionsbereiche "**Entdecke Meere**", "**Quiz**" und "**Puzzle**" prominent im oberen Bereich platziert und mit passenden Hintergrundbildern hervorgehoben. Die Navigation zu den spezifischen **Meeresregionen** wurde in den unteren Bereich verlegt. Das visuelle Grundthema ist ein **dunkelblauer Hintergrund**, der die Ozeantiefe widerspiegelt. Die Puzzle-Seite erhielt ebenfalls ein eigenes Hintergrundbild.

In dieser Phase wurden erste **offene Aufgaben** für die Ozean-Detailseite (PE20) identifiziert: die Integration eines passenden Hintergrunds und die Einbindung der Tierbilder. Es wurde bereits die erste **Funktionseinschränkung** des Puzzle-Moduls (PE40) vermerkt.

Phase 2: Datenstruktur und erste Fehleranalyse

Der Fokus lag auf der **Datenstrukturierung** und der Fehlerbehebung an den Detailseiten. Die **zentrale JSON-Datenbank** wurde intensiv bearbeitet, wobei alle relevanten Tiere mit eindeutigen IDs, detaillierten Informationen und zugehörigen Fotodateien strukturiert und eingeordnet wurden.

Trotz dieser Datenpflege traten weiterhin **Fehlfunktionen** auf:

- Die Ozean-Detailseite (PE20) zeigte die eingebundenen **Tierbilder** aus der JSON-Datei nicht korrekt an.
- Die Puzzle-Seite (PE40) zeigte ebenfalls erhebliche Mängel, da weder die **Bilder** für die Puzzleteile noch die **zugehörigen Tierinformationen** geladen oder dargestellt wurden.

Phase 3: Pfadanpassungen und Korrekturen

Die Arbeit konzentrierte sich auf die Korrektur der **Bildpfade** und die Anpassung der Logik in den JavaScript-Dateien. Mit Unterstützung des Dozenten wurden die Pfade und die zugehörige Logik erfolgreich überarbeitet, da die Seite **PE40** die Bilder nicht korrekt anzeigen. Ähnliche Korrekturen wurden auch im Bereich des **Quiz-Moduls** vorgenommen.

5.2 Testdurchführung & Ergebnisse

Die Tests wurden als **Modultests** (Prüfung einzelner Komponenten) und **Integrationstests** (Prüfung der Schnittstellen zwischen Komponenten und Service) durchgeführt.

- **Test-Tool:** Manuelle Funktionsprüfung im Browser (Chrome/Firefox).
- **Umgebung:** Angular Development Server (`ng serve`) auf <http://localhost:4200/>.
- **Codebasis:** Letzter Commit der `main`-Branch des GitHub-Repositories [Adil-si/Earth-Oceans-Lerning-App](#).

Testtabelle:

ID	Testfall	Erwartetes Ergebnis	Status
TF01	App-Start	Startseite lädt korrekt	OK
TF02	Ozean-Auswahl	Detailseite wird angezeigt	OK
TF03	Quiz-Logik	Richtige Antwort = Grün, falsche = Rot	OK
TF04	Puzzle-Funktion	Puzzle lädt und funktioniert	Fehler

5.3 Zusammenfassende Bewertung

Die Testdurchführung bestätigt die **funktionale Stabilität von 80%** der Hauptmodule (PE10, PE20, PE30). Die modulare Architektur und die Datenzentrale (`OceanService`) arbeiten zuverlässig.

Das Modul **PE40 (Puzzle)** ist der einzige Bereich, der die Anforderungen **nicht vollständig erfüllt**. Der zugrundeliegende Fehler liegt in der komplexen logischen Interaktion zwischen den `swapPieces`-Methoden und der korrekten Aktualisierung des `Signals` für das leere Puzzleteil, was das Spiel unspielbar macht.

Der Aufwand für das **Debugging** des Puzzles wird als **Investition in tiefes technologisches Verständnis** (Lessons Learned) bewertet, auch wenn die finale Lösung noch aussteht. Das Projekt ist somit unter dem Vorbehalt des Mangels in PE40 abgabereif.

6. Wirtschaftlichkeits

Etrachtung

Die Betrachtung gliedert sich in einen qualitativen Nutzenvergleich (Soll vs. Ist) und eine quantitative Analyse der Aufwands Abweichung.

Ziel (Soll)	Status (Ist)	Abweichung	Begründung/Auswirkung
PE10: Home & Routing	Erreicht	Keine	Die Basisstruktur ist stabil und modular (Standalone Components).
PE20: Ozean-Details	Erreicht	Keine	Datenanbindung und Darstellung der Fakten erfolgreich implementiert.
PE30: Quiz-Modul	Erreicht	Keine	Quiz-Logik ist vollständig funktionsfähig und auswertbar.
PE40: Schiebepuzzle	Funktion ist fehlerhaft	Funktionalität (Major)	Die swapPieces - and isAdjacent -Logik weist noch Fehler auf, die das Spielerlebnis verhindern. Die Komponente ist vorhanden, aber nicht nutzbar.
Gesamtnutzen	Mittel	Funktionalität	Der hohe pädagogische Nutzen der App ist durch die fehlende Funktionalität des Puzzles (PE40) aktuell noch stark beeinträchtigt.

6.1 Fazit Scope:

Die **Basisstruktur** und die linearen Module (PE10-PE30) entsprechen den Sollvorgaben. Das **Schlüssel-Feature PE40 (Puzzle)** ist der zentrale **Risikofaktor**, der aktuell die Wirtschaftlichkeit mindert, da der erwartete Gamification-Nutzen noch nicht eingelöst wird.

6.2 Soll-Ist-Vergleich des Aufwands:

Dieser Vergleich betrachtet die Abweichung zwischen dem geschätzten (geplanten) Aufwand und dem tatsächlich angefallenen Aufwand (IST).

Aufwandsbereich	SOLL (Plan-Std.)	IST (Tats. Std.)	Abweichung (\pm)	Bewertung
Basis & Daten	[25 Std.]	[25 Std.]	0 Std.	Plan erfüllt.
Quiz-Modul (PE30)	[20 Std.]	[20 Std.]	0 Std.	Plan erfüllt.
Puzzle-Modul (PE40)	[30 Std.]	[36 Std.]	+6 Std.	Überzogen
Dokumentation & Refactoring	[45 Std.]	[45 Std.]	0 Std.	Plan erfüllt.
Gesamtaufwand	[Gesamtsumme SOLL]	[Gesamtsumme IST]	[Gesamtabweichung]	Hauptabweichung liegt im Debugging.

6.3 Gesamtbewertung der Wirtschaftlichkeit:

Kriterium	IST-Situation	Auswirkungen auf die Wirtschaftlichkeit
Nutzwert	Die App ist inhaltlich bereit .	Aktuell geringer als geplant , da das Herzstück der Gamification (Puzzle) defekt ist.
Risiko	Das technische Risiko ist durch die Behebung kritischer Angular Fehler (NG 0203, Signal-Updates) stark reduziert.	Das verbleibende Risiko ist rein logischer Natur (ist Adjacent/swap Pieces).
Effizienz/Lerneffekt	Die zur Fehlersuche verlorene Zeit mindert zwar die Effizienz, hat aber einen hohen individuellen Lerneffekt erzeugt.	Die Auseinandersetzung mit der Komplexität hat tiefgreifendes Wissen über Change Detection und Signale vermittelt.

Die wirtschaftliche Tragfähigkeit des Projekts ist **potenziell hoch**, da eine wartbare, moderne Lernanwendung geschaffen wurde. Die **verlorene Zeit durch das Debugging** und die aktuell **fehlende Funktion des Puzzles (PE40)** mindern zwar die Effizienz, doch muss dieser Aufwand auch als **Investition in die eigene Qualifikation** betrachtet werden.

Perspektive des Entwicklers: "Als Schüler habe ich viele von diesen Fehlern in PE40 gelernt." "Ich gebe nicht auf und werde nach der Dokumentation sofort nach einer Lösung suchen, denn Fehler zu korrigieren und zu verstehen macht irgendwie Spaß."

Die Priorität muss nun auf der **Behebung des PE40-Fehlers** liegen, um den vollen Nutzen zu gewährleisten . Die Motivation, die **logische Herausforderung** des Puzzles zu meistern, ist gegeben und wird zur finalen Fertigstellung des Moduls führen.

7. Fazit & Ausblick

7.1 Zusammenfassung

Das Projekt zur Entwicklung der "**Earth-Oceans-Learning-App**" hat erfolgreich eine moderne, interaktive Lernanwendung auf Basis von **Angular Components** geschaffen. Die Kernziele der spielerischen Wissensvermittlung (Gamification) durch das **Quiz-Modul (PE30)**, die **Detailansichten (PE20)** und die **modulare Architektur** wurden erfüllt. Trotz der Herausforderungen im Bereich des **Puzzle-Moduls (PE40)** liefert die App eine solide technologische Basis für zukünftige Erweiterungen und stellt eine wertvolle Lernressource dar.

7.2 Lessons Learned

Dieser Abschnitt reflektiert die wichtigsten Erkenntnisse aus dem Projektverlauf in Bezug auf Management, Strategie und Technologie.

Projektmanagement & Strategie

Risiko „Bleeding Edge“: Die Entscheidung, auf moderne, teils noch junge Technologien wie **Angular Signals** zu setzen, führte zu erhöhten Debugging-Zeiten (insbesondere bei PE40), da Standardlösungen aus älteren Angular-Vor Versionen 14 nicht direkt übertragbar waren.

Erkenntnis: Das frühzeitige Eingehen solcher Risiken bietet zwar einen **Wettbewerbsvorteil** in Bezug auf Performance und Wartbarkeit, muss aber mit einem **Puffer im Zeitplan** kalkuliert werden.

Dokumentation : Die fortlaufende und detaillierte Dokumentation von Debugging-Sitzungen (wie bei der Behebung des **NG 0203-Fehlers** oder der **ID-0-Problematik**) war entscheidend.

Erkenntnis: Dieses Journaling spart später Zeit beim Nachvollziehen komplexer Logik Pfade und ist essentiell für die Projektabnahme.

Technische Erkenntnisse & Refactoring

Trennung von Zuständigkeiten: Die konsequente Trennung der **reinen Datenhaltung** (in **OceanService**) von der **komplexen Logik** (in **PuzzleComponent** und **QuizComponent**) hat die Komplexität jedes einzelnen Moduls beherrschbar gehalten.

Erkenntnis: Dieses Designprinzip erhöht die **Testbarkeit** und erleichtert den Austausch einzelner Module.

Mein erster, fehlerhafter Ansatz war, die Positionen der Puzzleteile direkt zu manipulieren (zu **mutieren**)

Das Problem, das ich nicht sehe:

Die von dir gezeigte Logik befüllt die Teile für das Puzzle **rein sequentiell** (**id: 1, 2, 3, ...**) und **ignoriert** die zuvor aus dem Service geladenen, **visuell korrekten** Teile (die Bilder/IDs des Tieres).

```
for (let i = 0; i < this.totalSlots; i++) {  
    initialParts.push({ id: (i === this.totalSlots - 1) ? 0 : i + 1,  
    // ID 0 ist das leere Feld // ... }); }
```

Konsequenz: Diese Schleife erstellt immer ein generisches 15-Puzzle (1, 2, 3, ... 15, 0), das **nichts mit dem ausgewählten Tierbild** zu tun hat. Die tatsächlichen Puzzleteile mit den IDs der Bilder wurden ignoriert

Die Korrektur: Fusion der Datenquellen

Um das Puzzle mit den **korrekten visuellen Inhalten** zu befüllen und gleichzeitig die **Situationslogik** (ID 0) beizubehalten, musst du die **puzzle Parts-Daten** (die visuellen/inhaltlichen IDs) mit deiner **Situationslogik** (dem leeren Feld) fusionieren.

Indem du die von dir gelieferte **puzzleParts**-Liste als **Basis** nimmst und nur das leere Feld (**id: 0**) **hinzufügst**, stellst du sicher, dass die Puzzleteile die **richtigen Tierbilder** darstellen.

Performance-Optimierung (Assets & LCP): Die Nutzung von **NgOptimizedImage** und die strategische Platzierung kritischer Assets (für den **Largest Contentful Paint - LCP**) sind für mobile Anwendungen unerlässlich.

Erkenntnis: Kleine Bildoptimierungen in der Startphase der App liefern sofort spürbare Performance-Vorteile.

7.3 Ausblick

Um den langfristigen Erfolg und die Reichweite der App zu sichern, sind folgende Erweiterungen geplant:

- **Barrierefreiheit & Text-to-Speech (Inklusion):** Implementierung von **ARIA-Attributen** und einer **Text-to-Speech (TTS)**-Funktion, um die App für Nutzer mit Behinderungen oder Lernschwierigkeiten zugänglicher zu machen.
 - **Internationalisierung (i18n):** Das Projekt soll durch die Einführung von i18n-Modulen (z.B. für Englisch und Arabisch) für eine **globale Zielgruppe** geöffnet werden. Dies erfordert die Auslagerung aller statischen Texte in Übersetzungsdateien.
 - **Erweiterung der Gamification:** Implementierung des neuen **Memory-Spiels (PE50)**, um das Angebot an interaktiven Modulen zu vergrößern und die Nutzerbindung zu erhöhen.
-

7.4 Projektabnahme und Übergabe

Die Projektabnahme erfolgt nach erfolgreicher Behebung des Fehlers im **Puzzle-Modul (PE40)**.

1. **Code-Überprüfung:** Der gesamte Code wird auf Einhaltung der Kodierungsrichtlinien (Linter, TypeScript-strikte Regeln) geprüft.
 2. **Übergabe der Dokumentation:** Übergabe des vollständigen Projekt-Journals und der technischen Dokumentation.
 3. Die formelle Abnahme des Projekts "**Earth-Oceans-Learning-App**" erfolgt nach dem folgenden Verfahren. Die Übergabe an Herrn Antonius Ahlen wird standardisiert durchgeführt, um die **Nachvollziehbarkeit** und **Wartbarkeit** des entwickelten Produktes sicherzustellen.
-

A. Anhang

A.0 Glossar:

- **Angular:** Das TypeScript-basierte Framework zur Entwicklung Ihrer Webanwendung.
- **SPA (Single Page Application):** Das Architekturprinzip der App; die Anwendung lädt nur eine HTML-Seite, Inhalte werden dynamisch nachgeladen.
- **Signals:** Das reaktive State-Management-System in Angular, das für die Zustandsverwaltung und die automatische Aktualisierung der UI (z.B. in Quiz und Puzzle) zuständig ist.
- **Tailwind CSS:** Das Utility-First CSS-Framework für das Styling und Layout der Benutzeroberfläche.
- **JSON (JavaScript Object Notation):** Das Datenformat, das zur Speicherung Ihrer gesamten Tier- und Ozean-Datenbank verwendet wird.
- **WCAG (Web Content Accessibility Guidelines):** Die internationalen Richtlinien, die für die geplante Barrierefreiheit und Inklusion (Text-to-Speech) relevant sind.
- **PE-Nummerierung:** Ihre interne Methode zur Projekt-Element Nummerierung (z.B. PE40), die zur Strukturierung der Dokumentation und des Soll-Ist-Vergleichs dient.
- **Gamification:** Das Prinzip des Einsatzes von Spielelementen (Quiz, Puzzle) in der Lern-App, um die Motivation der Nutzer zu steigern.
- **Injection Kontext:** Ein kritischer, Angular-spezifischer Kontext, dessen korrektes Verständnis für die Behebung des NG 0203-Fehlers notwendig war.
- **Immutable Update:** Ein zentrales State-Management-Prinzip. Es besagt, dass bei Zustandsänderungen immer ein neues Objekt zurückgegeben werden muss, anstatt das alte direkt zu verändern – dies war die Fehlerquelle in der Puzzle-Logik.

A.1 Quellenverzeichnis:

- Angular Dokumentation: <https://angular.dev>
- Tailwind CSS: <https://tailwindcss.com/docs>
- WCAG: <https://www.w3.org/WAI/standards-guidelines/wcag/>
- Mermaid.JS: Mermaid lets you create diagrams <https://mermaid.js.org/intro/>

A.1 Verzeichnisstruktur

```
Earth-Oceans-Lerning-App/
├── .vscode/                      # Visual Studio Code Einstellungen
├── public/                        # Öffentliche Assets (z.B. Favicon, statische Bilder)
└── src/                           # ↵ HAUPT-ANWENDUNGSCODE
    ├── app/                         # Die gesamte Angular-Anwendung
    │   ├── app.config.ts             # Globale Anwendungskonfiguration
    │   ├── app.routes.ts            # ↵ Routing-Definition (PE10 – PE40)
    │   ├── app.component.ts         # Hauptkomponente (Layout, Header/Footer)
    │   ├── home/                   # PE10: Home-Ansicht & Navigation
    │   └── page/                   # ↵ SPEZIFISCHE SEITEN-MODULE
    │       ├── ocean/              # PE20: Ozean-Detailseite (Datenansicht)
    │       ├── quiz/               # PE30: Quiz-Modul (QuizComponent & Logik)
    │       ├── puzzle/              # PE40: Puzzle-Modul (PuzzleComponent & fehlerhafte Logik)
    │       └── memory/              # PE50: (Geplante) Memory-Game-Komponente
    │       └── datatyps/             # Interfaces und Typdefinitionen (z.B. Animal, PuzzleData)
    │           └── service/          # ↵ DATENZENTRALE
    │               └── ocean.service.ts # Datenabruf und Geschäftslogik (PE20, PE30, PE40)
    ├── assets/                       # Bilder, Fonts und andere Ressourcen
    ├── environments/                # Umgebungsvariablen (prod/dev)
    ├── main.ts                      # Startdatei der Angular-Anwendung
    └── styles.css                  # Globale Stile
├── .editorconfig                 # Editor-Einstellungen
├── .gitignore                    # Dateien, die von Git ignoriert werden sollen
├── README.md                     # ↵ LIES MICH (Installationsanweisungen)
├── angular.json                  # Angular CLI Konfiguration
├── package.json                  # Projekt-Metadaten und Abhängigkeiten
└── tsconfig.json                 # TypeScript Compiler-Einstellungen
```

A.3 Code

Routing-Konfiguration

```
export const routes: Routes = [
  // { path: '', component: HomeComponent, title: 'Home' }, // PE10
  { path: '', loadComponent: () => import('./home/home.component').
    then(m => m.HomeComponent), title: 'Home' }, // PE10
  { path: 'ocean/:id', loadComponent: () =>
    import('./page/ocean/ocean.component').
    then(m => m.OceanComponent), title: 'Ozean Detail' }, // PE20
  { path: 'quiz', loadComponent: () => import('./page/quiz/quiz.component').
    then(m => m.QuizComponent), title: 'Quiz' }, // PE30
  { path: 'puzzle/:id', loadComponent: () =>
    import('./page/puzzle/puzzle.component').
    then(m => m.PuzzleComponent), title: 'Tier Puzzle' }, // PE40
  { path: '**', redirectTo: '' }
];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

PuzzleComponent: Logik

```
@Component({
  selector: 'app-puzzle',
  standalone: true,
  templateUrl: './puzzle.component.html',
  styleUrls: ['./puzzle.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush,
  imports: []
})
export class PuzzleComponent {
  // Dependency Injection (Services)
  private route = inject(ActivatedRoute);
```

```

private oceanService = inject(OceanService);
private router = inject(Router);
public puzzleData = toSignal(
  this.route.paramMap.pipe(
    map(params => {
      const idString = params.get('id');
      if (!idString){
        return null;
      }
      return Number(idString);
    }),
    switchMap(id => {
      if(id == null || id === 0){
        return [undefined];
      }
      return [this.oceanService.getpuzzledata(id)];
    })
  )
);

```

```

public gameState = signal<ComponentPuzzleData | undefined>(undefined);
// Metadaten für das Gitter-Layout
public gridSize: number = 0; // z.B. 4 (für 4x4)
private totalSlots: number = 0; // z.B. 16 (4*4)
private hintTimer: any; // Speichert den Timer für den
private loadGameEffect = effect(() => {
  const serviceData = this.puzzleData();
  if (serviceData) {
    this.initializeGame(serviceData);
  }
}, { allowSignalWrites: true });

```

```

private initializeGame(data: PuzzleData) {
  if (this.hintTimer) clearInterval(this.hintTimer);

  const totalParts = data.animal.puzzleParts;
  this.totalSlots = totalParts + 1; // Puzzle-Teile + 1 für das Leere Feld
  this.gridSize = Math.sqrt(this.totalSlots); // Kantenlänge des Quadrats

  if (this.gridSize % 1 !== 0) {
    console.error("Ungültige Anzahl von Puzzleteilen für ein
quadratisches Gitter.");
    return;
  }
}

```

```

// Erzeugt die Teile mit korrekter und initialer Position
let initialParts: GamePart[] = [];
for (let i = 0; i < this.totalSlots; i++) {
    initialParts.push({
        id: (i === this.totalSlots - 1) ? 0 : i + 1, // ID 0 ist das
Leere Feld
        currentPosition: i,
        correctPosition: i,
        isHighlighted: false
    });
}

this.shuffle(initialParts);

// Setzt den initialen Spielzustand im Signal
this.gameState.set({
    animal: data.animal,
    parts: initialParts,
    isComplete: false
});

// Startet den Timer für den Hinweis
this.startHintTimer();
}

/**
 * Mischt die Puzzleteile zufällig.
 */
private shuffle(parts: GamePart[]): void {
    for (let i = parts.length - 2; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [parts[i], parts[j]] = [parts[j], parts[i]];
    }
    parts.forEach((part, index) => part.currentPosition = index);
}
// -----
// AF02: HauptLogik: Bewegen eines Teils
// -----

/**
 * Verschiebt ein angeklicktes Teil, falls es neben dem Leeren Feld liegt,
 * und prüft auf Sieg.
 */
movePiece(pieceId: number): void {
    const currentState = this.gameState();
    if (!currentState || currentState.isComplete) return;

```

```

const clickedPiece = currentState.parts.find(p => p.id === pieceId);
const emptySlot = currentState.parts.find(p => p.id === 0);

if (!clickedPiece || !emptySlot || pieceId === 0) return;

const clickedIndex = clickedPiece.currentPosition;
const emptyIndex = emptySlot.currentPosition;

// Prüfen, ob der Zug erlaubt ist (angrenzend)
if (this.isAdjacent(clickedIndex, emptyIndex)) {

    // 1. **ALLES IN EINEM UPDATE ZUSAMMENFASSEN**
    this.gameState.update(state => {
        if (!state) return undefined;

        // KORREKTUR DER SWAP-LOGIK (Unveränderlich)
        const currentParts = state.parts;

        const partA = currentParts.find(p => p.currentPosition ===
clickedIndex)!;
        const partB = currentParts.find(p => p.currentPosition ===
emptyIndex)!;

        // Neue Objekte mit getauschten Positionen
        const newPartA = { ...partA, currentPosition: emptyIndex,
isHighlighted: false };
        const newPartB = { ...partB, currentPosition: clickedIndex,
isHighlighted: false };

        // Neues Array erstellen
        const newParts = currentParts.map(p => {
            if (p.id === partA.id) return newPartA;
            if (p.id === partB.id) return newPartB;
            // Highlights entfernen
            return { ...p, isHighlighted: false };
        });
    });
}

```

Quiz Component: Methoden

```

@Component({
  selector: 'app-quiz',
  standalone: true, // Standalone
  templateUrl: './quiz.component.html',
  styleUrls:'./quiz.component.css',

  changeDetection: ChangeDetectionStrategy.OnPush,
  imports: [NgOptimizedImage]
}

```

```

})
export class QuizComponent implements OnDestroy {
  public qService = inject(QuizService);
  public router = inject(Router);
  public autoAdvanceTimeout: ReturnType<typeof setTimeout> | null = null;

  public answerGiven = signal(false);
  public isCorrect = signal(false);
  public autoContinue = signal(false);

  public quizService = this.qService;

  checkAnswer(userChoice: boolean): void {
    if (this.answerGiven()) return;
    this.answerGiven.set(true);
    const question = this.quizService.currentQuestion();
    const isCorrect = question.correctAnswer === userChoice;
    this.isCorrect.set(isCorrect);

    // Automatisch fortfahren bei richtiger Antwort (AF75/85)
    if (isCorrect) {
      this.autoContinue.set(true);
      this.startAutoAdvance();
    } else {
      this.autoContinue.set(false);
      this.clearAutoAdvance();
    }
  }

  nextQuestion(): void {
    this.clearAutoAdvance();
    this.quizService.moveToNextQuestion();
    this.answerGiven.set(false);
    this.isCorrect.set(false);
    this.autoContinue.set(false);
  }

  getButtonClass(buttonValue: boolean, question: QuizQuestion): string {
    if (!this.answerGiven()) return '';
    let classes = [];

    // Rot, wenn falsch geantwortet wurde
    if (buttonValue === !this.isCorrect() && this.answerGiven()) {
      classes.push('wrong-answer');
    }
    // Grün, für die korrekte Antwort
    if (buttonValue === question.correctAnswer) {
      classes.push('correct-answer');
    }
  }
}

```

```

    }
    return classes.join(' ');
}

goHome(): void { this.router.navigate(['/']); } // AF09
resetAndGoHome(): void {
    this.quizService.resetQuiz();
    this.clearAutoAdvance();
    this.goHome();
}

ngOnDestroy(): void {
    this.clearAutoAdvance();
}

public startAutoAdvance(): void {
    this.clearAutoAdvance();
    this.autoAdvanceTimeout = setTimeout(() => this.nextQuestion(), 4000);
}

public clearAutoAdvance(): void {
    if (this.autoAdvanceTimeout) {
        clearTimeout(this.autoAdvanceTimeout);
        this.autoAdvanceTimeout = null;
    }
}
}

```

QuizService: Datenbindung

```

export interface QuizQuestion {
    id: number;
    question: string;
    imagePath: string;
    correctAnswer: boolean;
    difficulty: 'medium' | 'hard';
    explanation: string;
}

```

```

interface AppDataQuiz { quizQuestions: QuizQuestion[]; }

@Injectable({ providedIn: 'root' })
export class QuizService {
  private http = inject(HttpClient);
  private currentQuestionIndex = signal(0);
  private questions = toSignal(
    this.http.get<AppDataQuiz>('daten.json').pipe(
      map(data => data.quizQuestions ?? [])
    ),
    { initialValue: [] as QuizQuestion[] }
  );

  public currentQuestion = computed(() =>
this.questions()[this.currentQuestionIndex()]);
  public isQuizFinished = computed(() => this.questions().length > 0 &&
this.currentQuestionIndex() >= this.questions().length);
  public moveToNextQuestion(): void {
    if (!this.isQuizFinished()) {
      this.currentQuestionIndex.update(index => index + 1);
    }
  }

  public resetQuiz(): void {
    this.currentQuestionIndex.set(0);
  }
}

```

HomeComponent: Navigation

```

@Component({
  selector: 'app-home',
  standalone: true,
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush,
  imports: [

```

```

    ]
})

export class HomeComponent {
  oceanService = inject(OceanService);
  router = inject(Router);

  public oceans = computed(() => this.oceanService.getOceans());

  goToOcean(oceanId: number): void { this.router.navigate(['/ocean',
  oceanId]); }
  goToQuiz(): void { this.router.navigate(['/quiz']); }
  goToPuzzle(): void { this.router.navigate(['/puzzle', 101]); }

  goToFirstOcean(): void {
    const firstOcean = this.oceans()[0];
    if (firstOcean) {
      this.goToOcean(firstOcean.id);
    }
  }

  buildOceanCardBackground(oceanId: number, fallback?: string): string {
    const image = OCEAN_CARD_BACKGROUNDS[oceanId] ?? fallback ??
    '/Bilder/05.jpg';
    return `linear-gradient(135deg, rgba(1, 12, 25, 0.8), rgba(3, 32, 58,
    0.6)), url(${image})`;
  }
}

```

OceanService: Methoden

```

@Component({
  selector: 'app-ocean',
  standalone: true,
  templateUrl: './ocean.component.html',

```

```
styleUrl: './ocean.component.css',
changeDetection: ChangeDetectionStrategy.OnPush,
imports: [NgOptimizedImage]
})
export class OceanComponent {
  private route = inject(ActivatedRoute);
  private oceanService = inject(OceanService);
  private router = inject(Router);

  public ocean = toSignal(
    this.route.paramMap.pipe(
      map(params => Number(params.get('id'))),
      switchMap(id => [this.oceanService.getOcean(id)])
    )
  );
  goToAnimalDetail(animalId: number): void {
    this.router.navigate(['/puzzle', animalId]);
  }
}
```

Erklärung zur Eigenständigkeit:

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ort, Datum: Neuss, 09.12.2025

©Adil Sathi