Angular Projektabschluss Sommer 2025

Fachinformatiker Anwendungsentwickler

Dokumentation zur Angular Projektarbeit

SmartyBear

Kinder Lern- und Prüf-Simulator Webapplication

Abgabe: Dortmund, den 29.08.2025

Prüfungsbewerber:

Marcel Neumann,

Marcel Alexandre,

Ausbildungsbetrieb:

M^2 Studio

Chaos-avanue 777

46 69 73 69 Entropy City

0800 / 19 13 2025

Inhaltsverzeichnis

1. Inhaltsverzeichnis

2. Einleitung

Viele Kinder nutzen digitale Lernangebote, um schulische Inhalte wie Mathematik, Deutsch, Englisch oder Logikaufgaben zu üben. Die aktuell eingesetzten Lern-Apps sind jedoch häufig nicht ausreichend auf die Bedürfnisse von Kindern zugeschnitten. Ziel des Projektes ist die Entwicklung einer webbasierten App, die es Kindern ermöglicht, Lerninhalte in individuellem Tempo zu üben und darüber hinaus eine spielerische Prüfungssimulation durchzuführen.

2.1 Projektumfeld

Das Projekt „SmartyBear – Kinder Lern- und Prüf-Simulator“wird im Rahmen der Ausbildung zum Fachinformatiker für Anwendungsentwicklung durchgeführt. Es handelt sich um ein praxisnahes Ausbildungsprojekt, das sowohl technische als auch didaktische Aspekte vereint. Ziel ist es, eine Softwarelösung zu schaffen, die sich an den Bedürfnissen von Kindern im Grundschulalter orientiert und gleichzeitig den Anforderungen an moderne Webanwendungen entspricht.

Das Projekt ist in einem schulischen und privaten Lernkontext angesiedelt. Kinder sollen spielerisch an den Umgang mit digitalen Lernumgebungen herangeführt werden, während Eltern und Lehrkräfte eine zusätzliche Unterstützung bei der Vermittlung von Wissen erhalten. Die Anwendung richtet sich dabei an Kinder im Alter von 4–10 Jahren und ist somit speziell auf die Anforderungen der Grundschule zugeschnitten.

Das Projektumfeld umfasst außerdem die Einbindung gängiger Technologien der Softwareentwicklung. So wird ein **Angular-Frontend** entwickelt, das mit einem **ASP.Net Core Backend** kommuniziert. Die Speicherung der Fragen, Antworten und Benutzerdaten erfolgt in einer **SQL-Datenbank**. Damit entspricht das Projekt auch den aktuellen Standards der Webentwicklung und bietet praxisnahe Einblicke in moderne Architekturen.

Die Umsetzung findet in einem begrenzten Zeitrahmen statt (ca. zwei Wochen), sodass der Fokus auf einer funktionalen Kernversion der Anwendung liegt. Erweiterungen wie Mini-Spiele oder ein ausgereiftes Gamification-System werden bewusst für zukünftige Projektphasen offengehalten.

Einleitung

2.2 Projektziele

Die Das zentrale Ziel des Projekts ist die Entwicklung einer webbasierten Lern- und Prüfungs-App, die Kindern auf spielerische und intuitive Weise schulische Inhalte vermittelt. Dabei sollen insbesondere folgende Ziele erreicht werden:

- **Bereitstellung von Fragenkatalogen** aus den Kernfächern Mathematik, Deutsch, Englisch und Logik.
- **Unterstützung mehrerer Fragetypen**: Multiple-Choice, Single-Choice sowie Eingabeaufgaben.
- **Zwei Lernmodi**:
    - *Übungsmodus* ohne Zeitlimit, bei dem Kinder in ihrem eigenen Tempo lernen können.
    - *Prüfungsmodus* mit Zeitlimit, der eine realistische Testsituation simuliert.
- **Kindgerechtes Feedback** durch Animationen, Belohnungen oder Punktesysteme, um die Motivation zu steigern.
- **Plattformunabhängigkeit**: Die App soll über den Browser auf PC, Tablet und Smartphone nutzbar sein.
- **Einfache Erweiterbarkeit**: Durch eine API-gestützte Architektur sollen Fragen und Inhalte flexibel austauschbar sein.

Ein weiteres Ziel ist die Stärkung der Medienkompetenz von Kindern, indem sie frühzeitig lernen, mit digitalen Lernumgebungen verantwortungsvoll umzugehen. Gleichzeitig sollen Eltern und Lehrkräfte eine Möglichkeit erhalten, Lernfortschritte nachzuvollziehen und die Kinder gezielt zu fördern.

Langfristig soll das Projekt als Grundlage für weitere Module dienen, etwa für den Ausbau von Gamification-Elementen oder die Integration zusätzlicher Fächer.

Einleitung

2.3 Projektabgrenzung

Da es sich um ein Ausbildungsprojekt mit begrenztem Zeitrahmen handelt, ist eine klare Abgrenzung der Inhalte erforderlich. Bestimmte Funktionen werden bewusst **nicht** in den Projektumfang aufgenommen, um die Komplexität zu reduzieren und die Kernziele sicherzustellen:

- **Keine Monetarisierung**: Es wird keine Integration von In-App-Käufen oder kostenpflichtigen Zusatzinhalten geben.
- **Keine Werbung**: Um die Kinder nicht abzulenken und ein sicheres Lernumfeld zu gewährleisten, wird bewusst auf Werbung verzichtet.
- **Keine langfristige Wartung**: Die Anwendung ist ein Prototyp für Ausbildungszwecke. Eine dauerhafte Betreuung oder Weiterentwicklung ist nicht Bestandteil dieses Projekts.
- **Keine native App-Umsetzung**: Zwar ist eine spätere Überführung in eine native Android- oder iOS-App denkbar, jedoch liegt der Fokus im Projekt ausschließlich auf der webbasierten Umsetzung.
- **Eingeschränkter Funktionsumfang**: Erweiterungen wie Mini-Spiele, soziale Interaktionsmöglichkeiten sind nicht Teil des initialen Projekts.

Die klare Abgrenzung stellt sicher, dass sich das Projektteam auf die wesentlichen Funktionen konzentriert: eine intuitive, funktionale und kindgerechte Lern- und Prüfungsanwendung.

2.4 Projektschnittstellen

Damit die Anwendung „SmartyBear“ ihre Funktionalität entfalten kann, ist eine klare Schnittstellendefinition zwischen Frontend und Backend erforderlich. Das Frontend (Angular) selbst enthält keine statischen Fragen oder Inhalte, sondern bezieht diese dynamisch über das Backend (ASP.Net Core Web-API).

3. Projektplanung

3.1 Projektphasen

Das Projekt wird in mehrere Phasen unterteilt:

- Projektdefinition und Planung
- Analysephase
- Entwurfs- und Implementierungsphase
- Qualitätsmanagement und Tests
- Dokumentation und Abnahme

3.2 Ressourcenplanung

Für die Umsetzung werden ein Entwicklungsrechner, Visual Studio Code, Jetbrains Raider, Angular CLI, SQL Server sowie gängige Browser für Tests benötigt. Als Versionsverwaltung wird GitHub eingesetzt.

Projektplanung

3.3 Zeitplanung

Die Projektdauer beträgt insgesamt rund 2 Wochen:

- Erstellung Pflichtenheft: 18.08.–19.08.2025
- Entwicklung: 20.08.–26.08.2025
- Test & Dokumentation: 27.08.–29.08.2025

Die Abgabe erfolgt am 29.08.2025.

Projektphase	**Geplante Zeit**
Projektdefinition	3h
Planung	12h
Projektdurchführung	42h
Qualitätsmanagement	11h
Projektdokumentation	12h
Projektabnahme	2h
Gesamt	**82h**

Projektplanung

3.4 Qualitätsplanung

Um die Qualität des Projekts sicherzustellen, wird ein strukturierter Testplan umgesetzt. Dieser orientiert sich an den zuvor definierten Anforderungen und deckt alle wesentlichen Aspekte der App ab. Ziel ist es, sicherzustellen, dass die Anwendung fehlerfrei funktioniert, benutzerfreundlich ist und den Erwartungen der Zielgruppe entspricht.

Geplante Maßnahmen zur Qualitätssicherung:

- **Unit-Tests:** Prüfung einzelner Funktionen und Komponenten auf korrekte Arbeitsweise.
- **Integrationstests:** Sicherstellen, dass die verschiedenen Module (Frontend, Backend, Datenbank) reibungslos zusammenspielen.
- **Usability-Tests:** Überprüfung, ob die Bedienung der App für Kinder verständlich und intuitiv ist.
- **Kompatibilitätstests:** Tests in gängigen Browsern (Chrome, Edge, Firefox, Safari), um plattformübergreifende Funktionalität sicherzustellen.
- **Abnahmetests:** Abgleich mit den Abnahmekriterien (Funktionsumfang, kindgerechtes Design, stabiler Ablauf der Szenarien).

Diese Maßnahmen stellen sicher, dass die Qualität nicht nur am Ende überprüft wird, sondern während des gesamten Entwicklungsprozesses berücksichtigt bleibt.

4. Analysephase

4.1 Ist-Analyse

Viele Lern-Apps sind entweder zu komplex oder zu oberflächlich. Kinder verlieren schnell die Motivation. Eltern wünschen sich eine App, die sowohl das Lernen als auch die Prüfungsvorbereitung unterstützt.

4.2 Soll-Konzept

SmartyBear soll eine einfache, spielerische und zugleich effektive Möglichkeit bieten, schulische Inhalte zu lernen. Es kombiniert Übungs- und Prüfungsmodus in einer kindgerechten Umgebung.

4.3 Zielgruppen & Nutzungsszenarien

Zielgruppe sind Kinder im Alter von 4–10 Jahren. Nutzungsszenarien sind:

- Eigenständiges Lernen
- Prüfungssimulation mit Zeitlimit
- Einsatz in Schulen oder Nachhilfe.

4.4 Kostenrechnung

Interne Kostenkalkulation – Projekt SmartyBear

Für die Durchführung des Projektes wurden interne Kosten kalkuliert. Grundlage bildet eine von der Geschäftsführung festgelegte **firmeninterne Pauschale von 12,00 € pro Stunde**. Diese deckt allgemeine Betriebskosten ab:

- Stromkosten
- Büromietkosten
- Arbeitsplatzkosten (Tische, Stühle, Ausstattung)

Analysephase

1. Herleitung des Azubi-Stundenlohns

Der kalkulatorische Stundenlohn pro Auszubildendem ergibt sich aus den jährlichen Ausbildungskosten, verteilt auf die Arbeitsstunden pro Jahr.

- Ausbildungsvergütung: 1.050 € pro Monat → 12.600 € pro Jahr
- Arbeitgeberanteile & Nebenkosten: ca. 1.200 € pro Jahr
- **Gesamtkosten pro Jahr: 13.800 €**
- Arbeitszeit: ca. 1.800 Stunden pro Jahr

13.800 € ÷ 1.800 h ≈ 7,67 € pro Stunde.
Unter Berücksichtigung von Ausbildungszeiten, Urlaub und Schulungen wird ein **kalkulatorischer Wert von 6,00 € pro Stunde** angesetzt.

Da zwei Auszubildende beteiligt sind, ergibt sich ein **Gesamtstundenlohn von 12,00 € pro Stunde**.

2. Kalkulatorischer Stundensatz

24,00 € pro Stunde (12,00 € Pauschale + 12,00 € Azubi-Kosten für 2 Personen)

Für die Abnahme durch den Projektleiter wird ein höherer Satz von **38,00 € pro Stunde** veranschlagt.

3. Sachmittelkosten

Neben den Personalkosten fallen einmalige Sachmittelkosten an:

- Serverkosten: 8,50 €
- Software-Lizenzen (IDE, DB-Tools, Tests): 50,00 €
- Hardware-Nutzung (anteilige Abschreibung, Entwicklungsrechner): 30,00 €

Gesamte Sachmittelkosten: 88,50 €

Analysephase

Kostenpunkt	**Zeit**	**Kosten / Stunde**	**Kosten (€)**
Projektdefinition	3	24,00 Euro	72,00 Euro
Planung	12	24,00 Euro	288,00 Euro
Projektdurchführung	42	24,00 Euro	1.008,00 Euro
Qualitätsmanagement	11	24,00 Euro	265,00 Euro
Projektdokumentation	12	24,00 Euro	288,00 Euro
Projektabnahme	2	38,00 Euro	76,00 Euro
Sachmittelkosten (einmalig)	-	-	88,50 Euro
Gesamtkosten	**82**		**2.084,50 Euro**

5. Entwurfsphase

5.1 Systemübersicht

Das Gesamtsystem setzt sich aus drei zentralen Komponenten zusammen: einem Angular-Frontend, einem ASP.NET Core Backend sowie einer SQL-Server-Datenbank. Jede dieser Schichten übernimmt dabei klar definierte Aufgaben und ist über Schnittstellen miteinander verbunden.

Das Frontend wurde mit dem Framework Angular realisiert und stellt die Benutzeroberfläche des Systems dar. Es ist für die Darstellung der Daten, die Interaktion mit dem Anwender sowie für die Eingabe und Validierung von Informationen verantwortlich. Durch den modularen Aufbau von Angular kann die Oberfläche flexibel erweitert und gewartet werden.

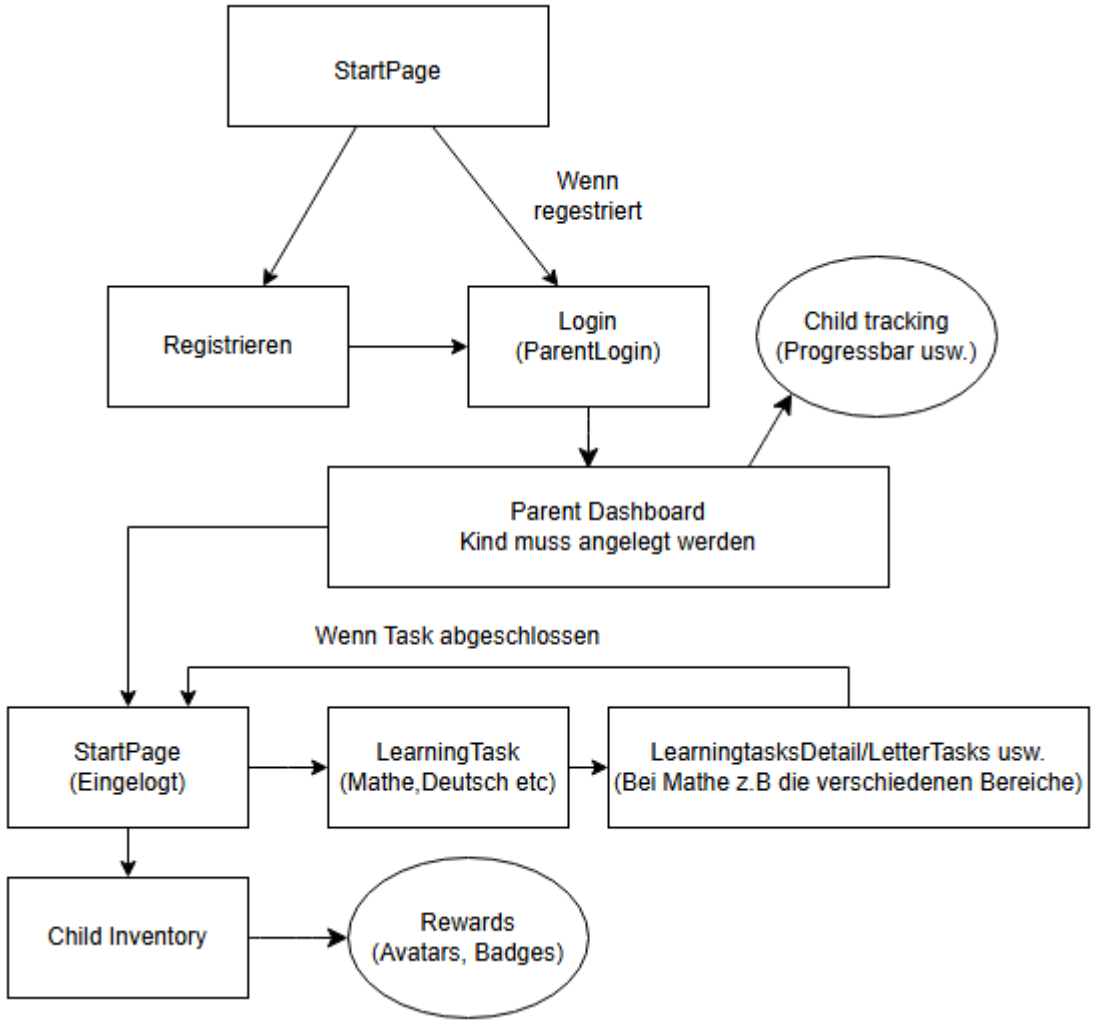
Die Kommunikation zwischen Frontend und Backend erfolgt über eine REST-API. Dabei werden standardisierte HTTP-Methoden (GET, POST, PUT, DELETE) eingesetzt, um Daten zu lesen, zu erstellen, zu ändern oder zu löschen.

Das Backend basiert auf ASP.NET Core und dient als zentrale Geschäftslogik-Schicht. Hier werden eingehende Anfragen verarbeitet, Validierungen durchgeführt und die Datenflüsse zwischen Frontend und Datenbank gesteuert. Zudem übernimmt das Backend Aufgaben wie Authentifizierung, Autorisierung und Fehlerbehandlung. Durch die klare Trennung der Logikschichten bleibt die Anwendung erweiterbar und wartungsfreundlich.

Die Datenhaltung erfolgt in einer SQL-Server-Datenbank. Dort werden sämtliche relevanten Informationen gespeichert. Das Backend greift über das Entity Framework Core auf die Datenbank zu, wodurch eine objektorientierte Abbildung der Datenmodelle ermöglicht wird. Dies reduziert den Aufwand für Datenbankabfragen und sorgt für eine saubere Trennung zwischen Daten- und Logikschicht.

Entwurfsphase

Aufbau Der App:



Entwurfsphase

5.2 Design & UX

Da es sich bei der Anwendung um eine Lernapp für Kinder im Alter von ca. 4 bis 10 Jahren handelt (Vorschule bis 4. Klasse), wurde das Design- und UX-Konzept speziell auf die Bedürfnisse dieser Zielgruppe abgestimmt. Der Schwerpunkt liegt dabei auf spielerischer Gestaltung, einfacher Bedienbarkeit und hoher Motivation.

Das Design ist bewusst farbenfroh, kindgerecht und visuell ansprechend gehalten. Große, klar erkennbare Buttons und leicht verständliche Symbole erleichtern die Bedienung auch für jüngere Kinder, die noch nicht sicher lesen können. Farben werden gezielt eingesetzt, um unterschiedliche Lernbereiche voneinander abzugrenzen und Orientierung zu schaffen. Zusätzlich sorgen Illustrationen und kindgerechte Grafiken für ein positives und einladendes Erscheinungsbild.

Das User Experience (UX) Konzept legt besonderen Wert auf Intuitivität und Motivation. Die Bedienung erfolgt nach dem Prinzip „so einfach wie möglich“, damit Kinder ohne lange Einarbeitung direkt starten können. Wiederkehrende Elemente, einheitliche Symbole und klare Navigationsstrukturen geben Sicherheit und fördern die Selbstständigkeit.

Um die Motivation beim Lernen zu erhöhen, werden spielerische Elemente integriert, wie z. B.:

Belohnungssysteme (Sterne, Punkte, Abzeichen)

Fortschrittsanzeigen (z. B. Level oder Lernfortschritt)

positive Rückmeldungen (z. B. Lobmeldungen, Animationen bei Erfolg)

Darüber hinaus wurde auf Barrierefreiheit geachtet. Große Schriftarten, deutliche Kontraste und Vorlesefunktionen (Audio-Ausgabe von Texten) unterstützen auch Kinder mit Lernschwierigkeiten oder eingeschränkten Lesefähigkeiten.

5.3 Technologien

Frontend: Angular 20 mit TypeScript

Das Frontend wurde mit Angular 20 entwickelt. Durch die komponentenbasierte Architektur und TypeScript lassen sich UI-Bausteine sauber trennen, wiederverwenden und effizient warten. Angular bietet integrierte Funktionen wie Routing, Formulare und Datenbindung, wodurch komplexe Benutzeroberflächen umgesetzt werden können.

Backend: ASP.NET Core Web-API

Das Backend basiert auf ASP.NET Core und stellt eine REST-API für den Datenaustausch bereit. Es bietet Modularität, Sicherheit und Performance sowie integrierte Funktionen für Authentifizierung, Autorisierung, Validierung und Logging.

Datenbank: SQL Server mit Entity Framework Core

Alle Benutzerdaten, Lerninhalte und Fortschrittsinformationen werden im SQL Server gespeichert. Entity Framework Core ermöglicht die objektorientierte Abbildung der Datenmodelle, vereinfacht Datenbankabfragen und unterstützt Migrationsprozesse.

Versionsverwaltung: GitHub

Der Quellcode wird über Git und GitHub verwaltet. GitHub dient dabei nicht nur der Versionskontrolle, sondern auch der Kollaboration, Nachverfolgbarkeit von Änderungen und Qualitätssicherung durch Pull Requests, Issue-Tracking und Projektboards.

5.4 Datenmodell

Die Lernapp verwendet eine relationale SQL Server-Datenbank, in der sämtliche relevanten Informationen über Benutzer, Kinder, Lerninhalte und Lernfortschritte gespeichert werden. Die Tabellen sind über Primär- und Fremdschlüssel miteinander verknüpft, wodurch Datenintegrität und konsistente Datenhaltung sichergestellt werden.

Für den Zugriff auf die Datenbank wird im Server-Backend das Entity Framework Core verwendet. Dieses erleichtert die Abbildung des objektorientierten Datenmodells auf die relationale Datenbank und ermöglicht eine effiziente und sichere Datenverwaltung.

6. Implementierungsphase

Die Implementierung der Lernapp erfolgte nach dem Entwurfskonzept, das in den vorangegangenen Kapiteln beschrieben wurde. Sie umfasst die Entwicklung des Frontend, des Backend sowie der Datenübertragungsschicht, um eine stabile und benutzerfreundliche Lernumgebung zu schaffen.

Frontend (Angular 20)

Im Frontend wurden verschiedene Angular-Komponenten erstellt, die die Benutzeroberfläche und die Interaktion mit den Lerninhalten realisieren

Die Komponenten sind modular aufgebaut und wiederverwendbar. Durch die Nutzung von Angular-Services wird die Kommunikation mit dem Backend zentralisiert, sodass Datenanfragen konsistent und wartbar sind.

Backend (ASP.NET Core Web-API)

Das Backend wurde als RESTful Web-API in ASP.NET Core umgesetzt. Es stellt die Geschäftlogik der Anwendung bereit und ist verantwortlich für:

- Verarbeitung von Anfragen aus dem Frontend
- Validierung von Benutzereingaben
- Authentifizierung und Autorisierung über ASP.NET Identity
- Verwaltung der Lerninhalte, Aufgaben, Badges und Fortschrittsdaten

Zur Datenübertragung zwischen Frontend und Backend werden DTOs (Data Transfer Objects) verwendet. Diese dienen dazu, nur die relevanten Daten an das Frontend zu liefern und gleichzeitig die internen Datenbankstrukturen zu kapseln. DTOs sorgen für Sicherheit, Performance und Klarheit in den API-Schnittstellen.

Implementierungsphase

Datenbankzugriff

Der Zugriff auf die SQL-Server-Datenbank erfolgt über Entity Framework Core. Die Implementierung umfasst:

- Abbildungen der Datenbanktabellen in C#-Klassen (Entities)
- CRUD-Operationen (Create, Read, Update, Delete) für alle relevanten Entitäten
- Implementierung von Relations- und Cascade-Regeln entsprechend dem Datenmodell

Die Projektschnittstellen lassen sich wie folgt beschreiben:

- **Frontend → Backend**
  Das Frontend stellt HTTP-Anfragen an die REST-API des Backends. Beispiele sind:
  - *GET /api/tasks* → liefert eine Liste von Fragen mit Antwortmöglichkeiten.
  - *POST /api/learning/{childDTO}* → überträgt die vom Benutzer eingegebenen Antworten.
  - *GET /api/learning/{childId}/{taskId}* → gibt das Ergebnis eines Tests zurück.
  - *GET /api/Tasks/all-questions:*


```
{
  "id": 562,
  "text": "Wähle die passende Form für das Muster",
  "correctAnswer": "assets/questImg/form.heart.png",
  "imageUrl": "assets/questImg/form.heartQuestion.png",
  "options": [
    "assets/questImg/form.label.png",
    "assets/questImg/form.heart.png",
    "assets/questImg/form.tear.png",
    "assets/questImg/form.new-moon.png"
  ],
  "learningTaskId": 12,
  "learningTask": null,
  "difficulty": "Vorschule",
  "category": ""
},
```

Implementierungsphase

- **Backend → Datenbank**
  Das Backend greift über Entity Framework Core auf die SQL-Datenbank zu. Dort werden die Fragen, Antwortoptionen, Ergebnisse und Benutzerinformationen gespeichert und verwaltet.

- **Frontend ← Backend**
  Nach der Abfrage stellt das Backend die Daten in Form von **DTOs (Data Transfer Objects)** bereit. Diese enthalten nur die für die jeweilige Abfrage relevanten Informationen. Das Frontend nutzt diese Daten, um die Fragen darzustellen, Ergebnisse zu berechnen und Feedback anzuzeigen.

Um die Nachvollziehbarkeit und Wartbarkeit zu gewährleisten, wurden Screenshots der Benutzeroberfläche (4)(5) und Codebeispiele (6) erstellt und im Anhang der Projektdokumentation dokumentiert. Dies ermöglicht eine direkte Einsicht in die Struktur und Funktionsweise der implementierten Komponenten und erleichtert die spätere Erweiterung oder Anpassung der App.

Zusammenfassung

Die Implementierungsphase stellt den Übergang von der konzeptionellen Planung zur funktionsfähigen Anwendung dar. Durch die Kombination aus modularem Angular-Frontend, leistungsfähigem ASP.NET Core Backend und sauberem Datenbankzugriff entsteht eine robuste, erweiterbare und kinderfreundliche Lernplattform.

7. Qualitätsmanagement

Das Qualitätsmanagement stellt sicher, dass die Lernapp stabil, zuverlässig und kindgerecht funktioniert. Zur Sicherstellung der Funktionalität und Usability wurden folgende Tests durchgeführt:

- Funktionstests: Überprüfung der Kernfunktionen, darunter korrekte Darstellung von Fragen und Aufgaben, Auswertung der Antworten, Aktualisierung des Belohnungssystems sowie fehlerfreie Datenbankinteraktionen.
- Usability-Tests: Beobachtung von Kindern beim Umgang mit der App, Überprüfung der Navigation, Buttons und Symbole sowie Sicherstellung, dass Lernmodule und Belohnungen selbsterklärend und motivierend sind. Feedback wurde genutzt, um Layout, Farben, Icons und Texte anzupassen.
- Performancetests: Sicherstellung schneller Reaktionszeiten (Ladezeiten < 3 Sekunden), Prüfung der API-Antwortzeiten und Simulation mehrerer gleichzeitiger Nutzer zur Stabilitätsprüfung.

Durch diese Maßnahmen wird gewährleistet, dass die App funktional, benutzerfreundlich und performant ist.

8. Abnahmephase

Die Abnahme des Projektes erfolgte anhand der zuvor definierten Abnahmekriterien. Im Rahmen dieser Prüfung wurde überprüft, ob die App sämtliche vorgesehenen Fragetypen korrekt unterstützt und fehlerfrei darstellt. Zudem wurde das Design eingehend bewertet, um sicherzustellen, dass es kindgerecht, ansprechend und intuitiv bedienbar ist. Weiterhin wurden alle Testszenarien durchgeführt, um die Funktionsfähigkeit der App in unterschiedlichen Nutzungssituationen zu bestätigen. Hierbei wurde geprüft, ob die Lerninhalte korrekt geladen werden, die Interaktionen zuverlässig funktionieren und mögliche Fehlermeldungen angemessen behandelt werden. Erst nachdem alle Abnahmekriterien erfolgreich erfüllt und die Qualitätsstandards des Projekts sichergestellt waren, erfolgte die endgültige Freigabe durch den Projektleiter.

9. Fazit

Das Projekt SmartyBear wurde erfolgreich umgesetzt und erfüllt die im Pflichtenheft definierten Anforderungen. Die App stellt eine kindgerechte Lern- und Prüfungsumgebung bereit, die speziell für Kinder von der Vorschule bis zur 4. Klasse konzipiert wurde. Durch die Kombination aus spielerischer Gestaltung, klarer Struktur und einfacher Bedienbarkeit können Kinder eigenständig lernen und ihren Lernfortschritt verfolgen.

Fazit

Soll-/Ist-Vergleich

Funktionalität:

Im Soll war vorgesehen, dass die App folgende Kernfunktionen enthält:

- Übungsmodus ohne Zeitlimit, um selbstbestimmtes Lernen zu ermöglichen
- Quizmodus mit Zeitlimit und automatischer Auswertung zur Selbstkontrolle
- Speicherung des Lernfortschritts im Backend
- Responsive Design für die Nutzung auf PC, Tablet und Smartphone

Im Ist wurden alle diese Funktionen erfolgreich umgesetzt. Zusätzlich wurde ein Belohnungssystem integriert, das Sterne, Abzeichen und Fortschrittsanzeigen umfasst, um die Motivation der Kinder zu steigern.

Abweichungen in der Funktionalität:

- Die geplante Integration von Mini-Spielen konnte aus Zeitgründen nicht realisiert werden. Diese Funktion ist für eine spätere Projektphase vorgesehen und stellt eine sinnvolle Erweiterung dar.

Zeit:

Die Umsetzung erfolgte im vorgesehenen Projektzeitraum von ca. zwei Wochen. Lediglich kleinere Verschiebungen ergaben sich durch die Integration zusätzlicher Features (Belohnungssystem). Der Meilensteinplan konnte jedoch insgesamt eingehalten werden.

Kosten:

Die kalkulierten Projektkosten lagen bei ca. **2.084,50** €. Dieser Rahmen wurde eingehalten, da überwiegend firmeneigene Ressourcen sowie Open-Source-Tools verwendet wurden. Zusätzliche ungeplante Kosten sind nicht entstanden.

Fazit

Qualität:

Die Qualitätssicherung wurde durch Funktionstests, Usability-Tests mit Kindern sowie Performancetests sichergestellt. Alle kritischen Anforderungen wurden erfüllt. Die App läuft stabil, ist benutzerfreundlich und erfüllt die Qualitätsstandards, die im Pflichtenheft und in den Testplänen definiert wurden.

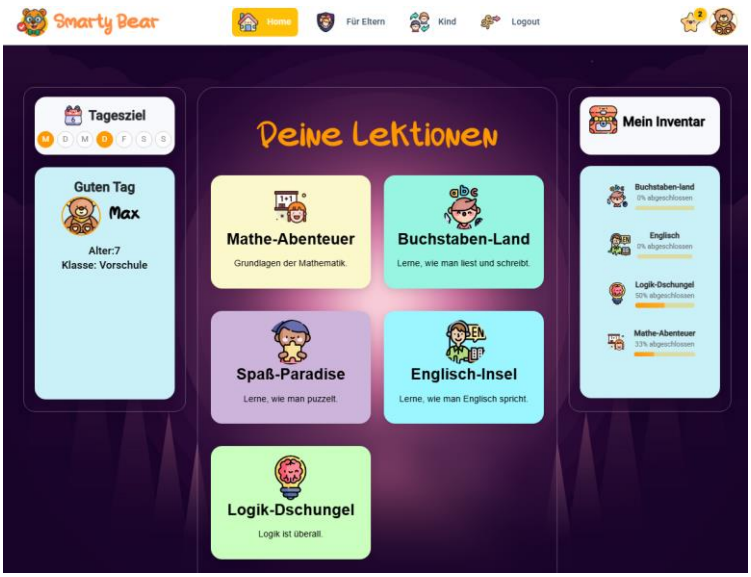
Zusammenfassung:

Die App ist funktional, stabil und einsatzbereit. SmartyBear erfüllt die wesentlichen Anforderungen des Pflichtenhefts und bietet eine solide Grundlage für zukünftige Erweiterungen. Potenzielle Ausbaumaßnahmen wie zusätzliche Gamification-Elemente, erweiterte Lerninhalte oder Analysefunktionen für Eltern und Lehrkräfte können in zukünftigen Versionen integriert werden. Insgesamt zeigt das Projekt, dass eine modulare, benutzerzentrierte Lernplattform für Kinder erfolgreich umgesetzt werden kann.

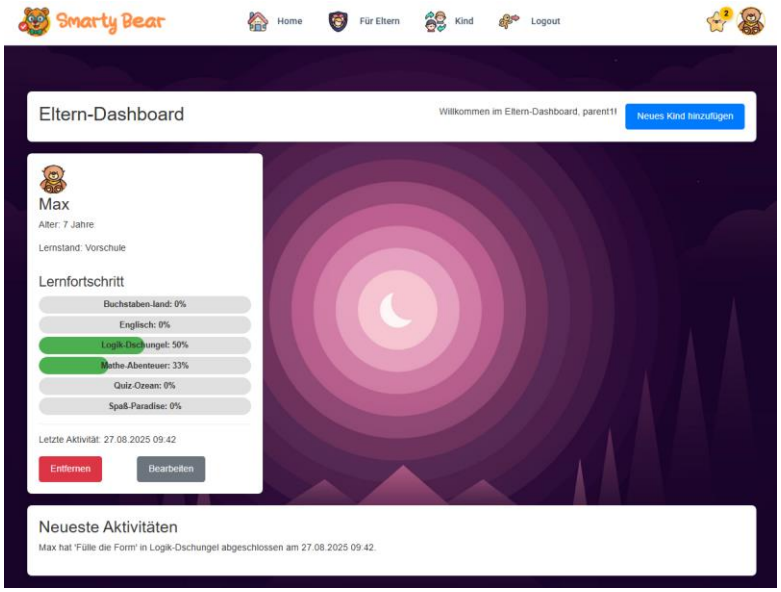
10. Anhang

4. A1: Screenshots der App

Screenshot 1: Startseite der App



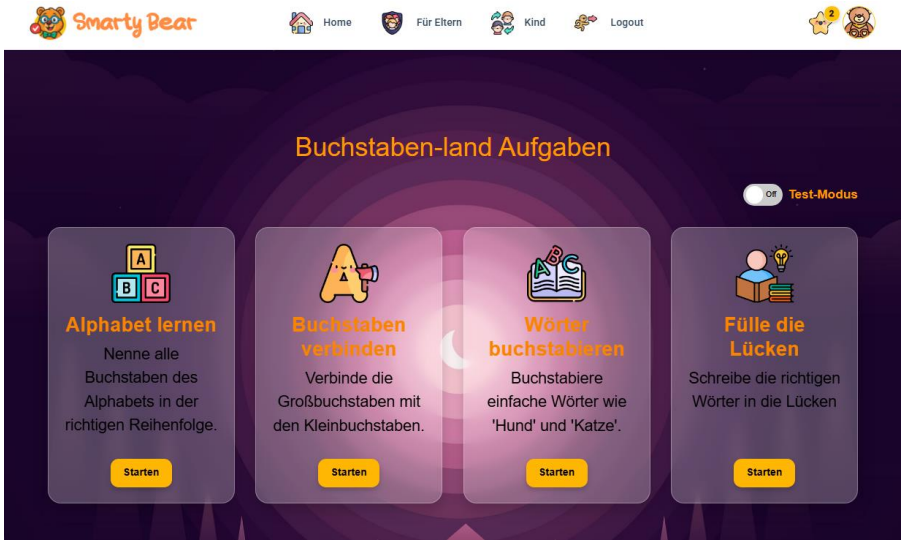
Screenshot 2: Eltern Dashboard



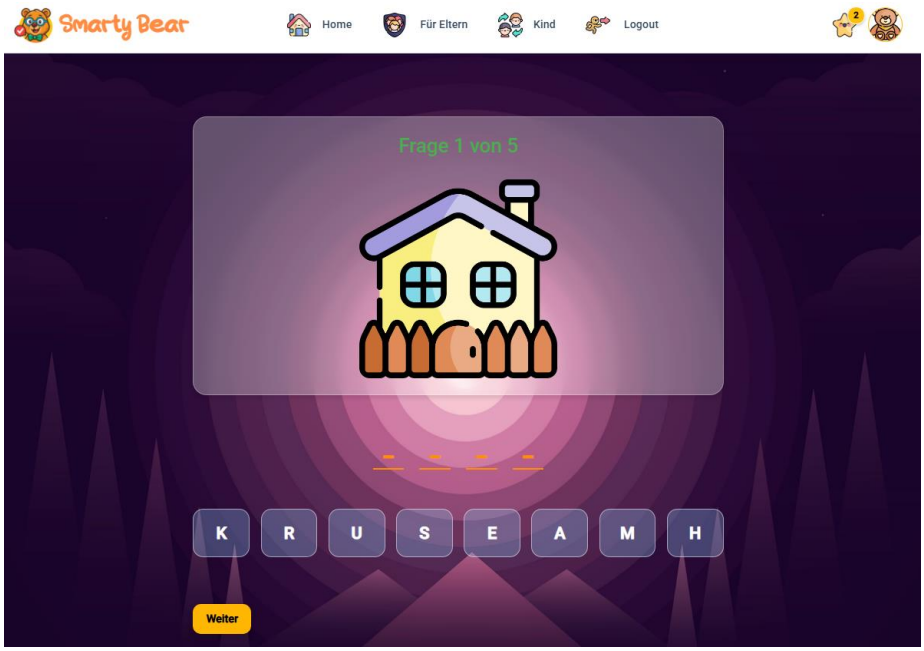
Anhang

5. A2: Screenshots der App

Screenshot 3: Aufgaben Auswahl



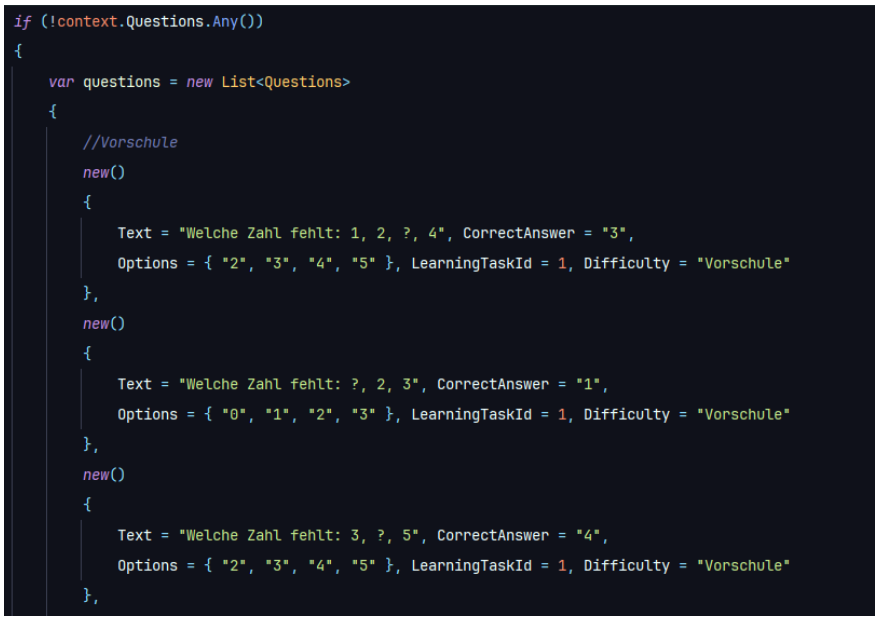
Screenshot 4: Beispiel Aufgabe



Anhang

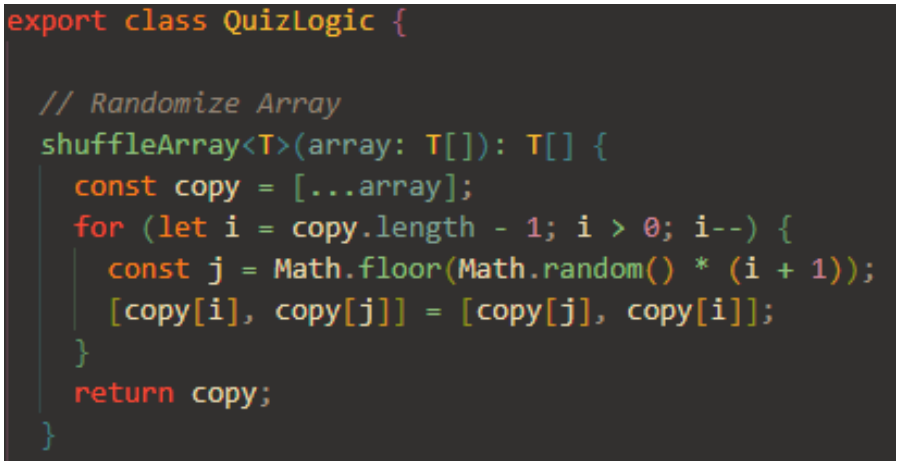
6. A3: Code Beispiele

Screenshot 5: Ausschnitt Seeding der Datenbank



```
if (!context.Questions.Any())
{
    var questions = new List<Questions>
    {
        //Vorschule
        new()
        {
            Text = "Welche Zahl fehlt: 1, 2, ?, 4", CorrectAnswer = "3",
            Options = { "2", "3", "4", "5" }, LearningTaskId = 1, Difficulty = "Vorschule"
        },
        new()
        {
            Text = "Welche Zahl fehlt: ?, 2, 3", CorrectAnswer = "1",
            Options = { "0", "1", "2", "3" }, LearningTaskId = 1, Difficulty = "Vorschule"
        },
        new()
        {
            Text = "Welche Zahl fehlt: 3, ?, 5", CorrectAnswer = "4",
            Options = { "2", "3", "4", "5" }, LearningTaskId = 1, Difficulty = "Vorschule"
        },
```

Screenshot 6: Funktion zum Fragen/Antworten Mischen



```
export class QuizLogic {

  // Randomize Array
  shuffleArray<T>(array: T[]): T[] {
    const copy = [...array];
    for (let i = copy.length - 1; i > 0; i--) {
      const j = Math.floor(Math.random() * (i + 1));
      [copy[i], copy[j]] = [copy[j], copy[i]];
    }
    return copy;
  }
```