QUIZ-APP

Semesterprojekt: Implementierung eines einfachen Games

EXP
Semesterarbeit, Quiz App

Dino Poljak & Eric Ferrari-Hermann
JPL, Java Project Level

FFHS
Fernfachhochschule Schweiz
Mitglied der SUPSI

Inhalt

1 Einführung und Ziele

Dieses Projekt besteht darin, eine Quiz-Anwendung zu entwickeln, die über ein Netzwerk läuft. Die Anwendung besteht aus einem Server und mehreren Clients. Der Server ist dafür verantwortlich, Quizfragen an die Clients zu senden und deren Antworten zu bewerten. Die Clients sind die Spieler, die das Quiz spielen.

Die Quizfragen werden vom Server aus einer Textdatei geladen. Jede Frage hat drei mögliche Antworten (A, B oder C), von denen genau eine richtig ist. Nachdem der Server eine Frage an die Clients gesendet hat, wartet er auf deren Antworten, bewertet sie und sendet ein Feedback zurück, ob die Antwort richtig oder falsch war und wie der aktuelle Punktestand ist. Am Ende des Quizzes sendet der Server eine Nachricht, die den finalen Punktestand enthält.

Die Clients nehmen die Fragen entgegen, zeigen sie dem Benutzer an und warten auf dessen Antwort. Die Antwort wird dann an den Server zurückgesendet.

1.1 Qualitätsziele

Korrektheit: Das System muss die Fragen korrekt an die Clients senden, die Antworten korrekt bewerten und den Punktestand korrekt berechnen. Jede Abweichung hiervon könnte zu einer unfairen Bewertung der Spieler führen.

Robustheit: Das System muss in der Lage sein, mit Netzwerkfehlern und unerwarteten Eingaben umzugehen. Wenn zum Beispiel ein Client während des Quizzes die Verbindung verliert, sollte das System dies erkennen und entsprechend reagieren, anstatt abzustürzen oder in einen undefinierten Zustand zu geraten.

Benutzerfreundlichkeit: Die Benutzerschnittstelle der Clients sollte einfach und intuitiv zu bedienen sein, damit die Spieler das Quiz problemlos spielen können. Die Fragen und Optionen sollten klar und verständlich angezeigt werden, und es sollte offensichtlich sein, wie man eine Antwort auswählt und absendet.

Effizienz: Da es sich um ein Netzwerkspiel handelt, sollte das System so effizient wie möglich sein, um Latenz zu minimieren und eine schnelle Reaktion zu ermöglichen. Insbesondere sollte der Server in der Lage sein, gleichzeitig mit mehreren Clients umzugehen, ohne dass dies seine Leistung beeinträchtigt.

Erweiterbarkeit: Es sollte möglich sein, das System leicht um neue Funktionen zu erweitern, zum Beispiel um neue Arten von Fragen hinzuzufügen oder um das Quiz für verschiedene Themenbereiche zu konfigurieren. Dies erfordert, dass der Code gut organisiert und verständlich ist, und dass er so modulartig wie möglich gestaltet ist.

1.2 Stakeholder

Rolle	Kontakt	Erwartungshaltung
Entwicklerteam	*Intern*	*Das Entwicklerteam ist verantwortlich für das Design, die Implementierung und das Testen der Anwendung. Sie erwarten klare Anforderungen und Rückmeldungen zu ihrer Arbeit.*
Auftraggeber	*Direkter Kontakt*	*Der Auftraggeber finanziert das Projekt und möchte ein erfolgreiches, pünktlich geliefertes Produkt. Er erwartet eine regelmäßige Kommunikation über den Projektfortschritt und möchte sicherstellen, dass das Produkt seinen Anforderungen entspricht.*
Systemadministrator	*Intern*	*Der Systemadministrator ist verantwortlich für den Betrieb des Servers, einschließlich der Installation der Anwendung, der Überwachung ihrer Leistung und der Behebung von Problemen. Er erwartet eine gut dokumentierte, zuverlässige und wartungsfreundliche Anwendung.*
Projektmanager	*Intern*	*Der Projektmanager koordiniert das Projekt und ist verantwortlich für die Einhaltung von Fristen und Budgets. Er erwartet regelmäßige Updates zum Fortschritt des Projekts und möchte sicherstellen, dass alle Stakeholder zufrieden sind.*
Benutzer (Spieler)	*Kundensupport/E-Mail/Forum*	*Die Benutzer sind diejenigen, die das Quiz spielen. Sie erwarten eine unterhaltsame, fehlerfreie und benutzerfreundliche Spielerfahrung.*

1.3 Randbedingungen

Die Randbedingungen für dieses Projekt sind wie folgt:

Technologieauswahl: Das Projekt wird in Java implementiert, was bedeutet, dass die Anwendung auf jeder Plattform ausgeführt werden kann, die eine Java-Laufzeitumgebung unterstützt.

Netzwerkanforderungen: Da die Quiz-Anwendung über ein Netzwerk läuft, muss eine ständige und stabile Internetverbindung vorhanden sein. Darüber hinaus müssen die Firewall-Einstellungen so konfiguriert sein, dass der Datenverkehr zum und vom Server-Port zulässig ist.

Datensicherheit: Obwohl dieses Projekt keine besonders strengen Sicherheitsanforderungen hat, sollten Grundprinzipien der Datensicherheit beachtet werden. Insbesondere sollte die Kommunikation zwischen Server und Client, wenn möglich, verschlüsselt werden, um die Gefahr von Man-in-the-Middle-Angriffen zu minimieren.

Leistung: Der Server muss in der Lage sein, gleichzeitig eine große Anzahl von Clients zu bedienen. Dies bedeutet, dass der Servercode effizient gestaltet und gut getestet werden muss.

Benutzerfreundlichkeit: Die Clients sollten eine intuitive und reaktionsschnelle Benutzeroberfläche haben. Darüber hinaus sollten sie so konzipiert sein, dass sie auf unterschiedlichen Betriebssystemen und Geräten gut funktionieren.

Entwicklungszeit: Da dies ein Projekt mit festgelegtem Budget und Zeitrahmen ist, muss der Umfang des Projekts und der Ansatz zur Implementierung so gestaltet werden, dass sie innerhalb dieser Einschränkungen realisierbar sind. Bei der Planung des Projekts sollten mögliche Risiken und Hindernisse berücksichtigt und entsprechende Maßnahmen geplant werden.

Erweiterbarkeit: Das Quizsystem sollte so konzipiert sein, dass es leicht erweitert und angepasst werden kann. Dies könnte zum Beispiel bedeuten, dass es möglich sein sollte, neue Fragetypen hinzuzufügen, die Benutzeroberfläche zu ändern oder das Scoring-System anzupassen.

Fehlertoleranz: Da Netzwerkverbindungen unzuverlässig sein können und Benutzer oft unerwartete Eingaben machen, muss die Anwendung in der Lage sein, solche Situationen zu erkennen und angemessen darauf zu reagieren. Fehler sollten so behandelt werden, dass sie den Benutzer so wenig wie möglich stören und so viele Informationen wie möglich für die Entwickler bereitstellen, um das Problem zu diagnostizieren und zu beheben.

2. Kontextabgrenzung

Die Kontextabgrenzung beschreibt die externen Schnittstellen des Systems und zeigt auf, wie das System in ein größeres Gesamtbild passt.

2.1 Systemgrenzen

In unserem Fall besteht das System aus drei Hauptteilen:

QuizServer: Dieser Teil der Anwendung ist dafür verantwortlich, die Quizfragen zu verwalten und sie an die Clients zu senden. Er nimmt auch die Antworten der Clients entgegen und bewertet sie.

QuizClient: Dieser Teil der Anwendung stellt die Benutzeroberfläche für die Spieler bereit. Er nimmt die Fragen vom Server entgegen und zeigt sie dem Benutzer an. Er nimmt auch die Antworten des Benutzers entgegen und sendet sie an den Server.

Quiz-Daten: Dies sind die Daten, die die Quizfragen und -antworten enthalten. Sie werden vom Server aus einer Textdatei geladen.

2.2 Externe Schnittstellen

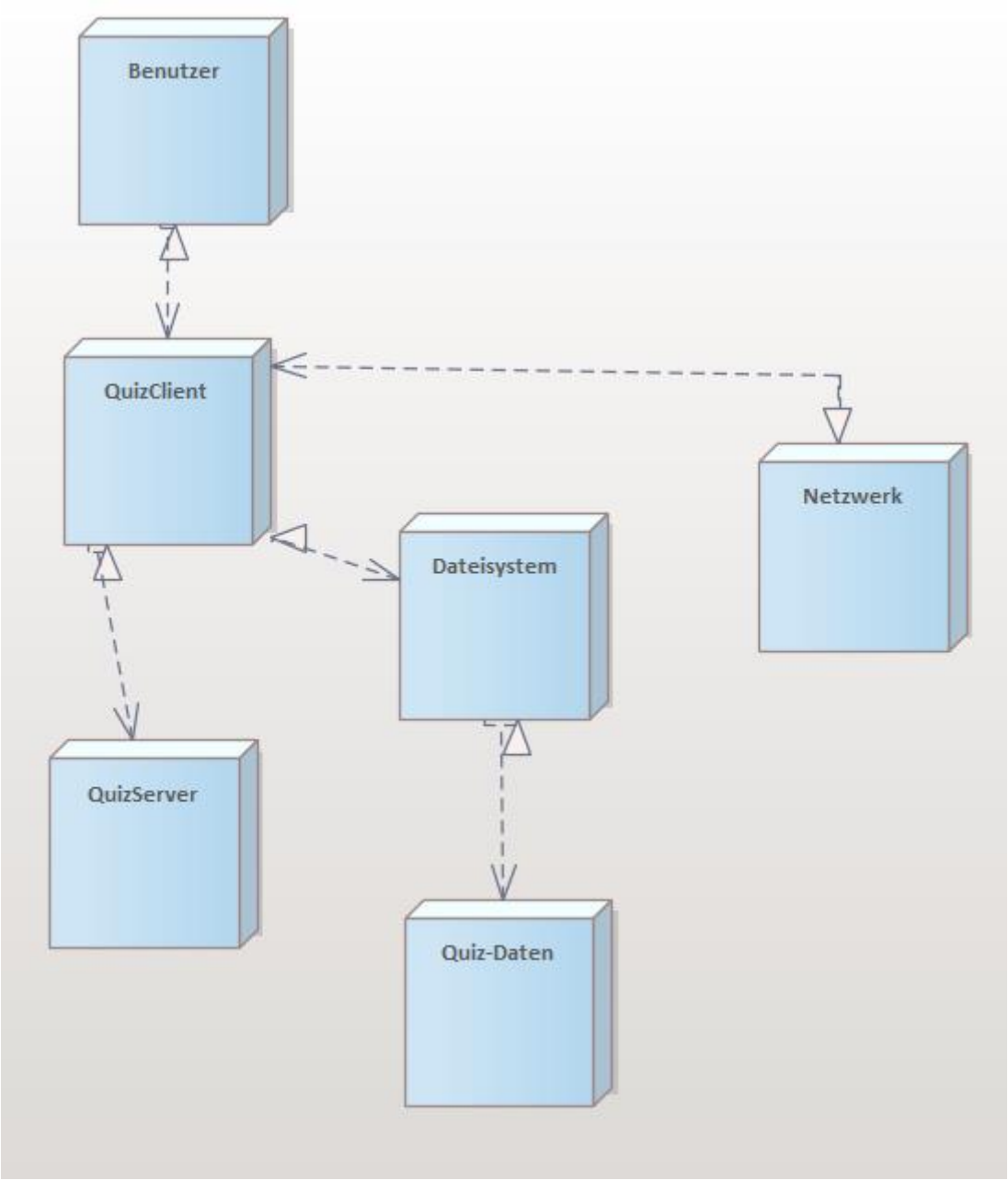
Das System interagiert über das Netzwerk mit seinen Clients und den Benutzern. Hier sind die wichtigsten Schnittstellen:

Netzwerkschnittstelle: Dies ist die Schnittstelle, die die Kommunikation zwischen dem Server und den Clients ermöglicht. Sie besteht aus den TCP-Sockets, die vom Server und den Clients geöffnet werden, und dem Protokoll, das verwendet wird, um die Fragen und Antworten zu senden und zu empfangen.

Benutzerschnittstelle: Dies ist die Schnittstelle, die der Benutzer verwendet, um das Quiz zu spielen. Sie besteht aus der Anzeige der Fragen und der Eingabe der Antworten.

Dateisystem: Dies ist die Schnittstelle, die der Server verwendet, um die Quizdaten zu laden. Sie besteht aus dem Dateisystem des Host-Computers und dem Dateiformat der Quizdaten.

2.3 Kontextdiagramm



In diesem Diagramm sind die Pfeile in die Richtung der Datenflüsse gerichtet. Zum Beispiel sendet der Benutzer Antworten an den Client, der Client sendet und empfängt Fragen und Antworten an und von dem Server über das Netzwerk, und der Server lädt die Fragen aus den Quiz-Daten über das Dateisystem.

3. Fachlicher und technischer Kontext

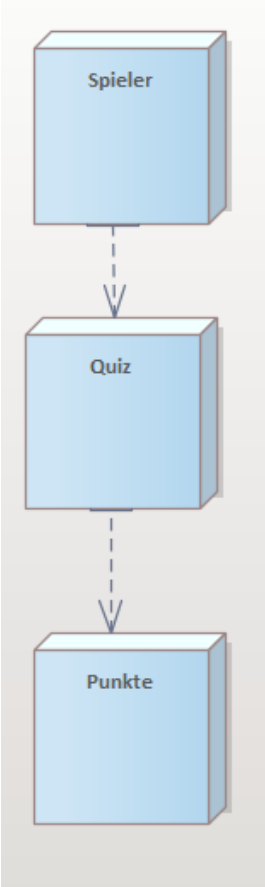
3.1 Fachlicher Kontext

Der fachliche Kontext betrifft die Geschäftslogik und die Regeln, die den Betrieb des Quiz-Systems steuern.

Quiz: Ein Quiz besteht aus einer Reihe von Fragen, die jeweils mehrere Antwortmöglichkeiten haben, von denen genau eine richtig ist.

Spieler: Ein Spieler interagiert mit dem Quiz-Client, um die Fragen zu sehen und die Antworten auszuwählen.

Punkte: Jede richtig beantwortete Frage erhöht den Punktestand des Spielers. Der Punktestand wird nach Beendigung des Quiz angezeigt.



3.2 Technischer Kontext

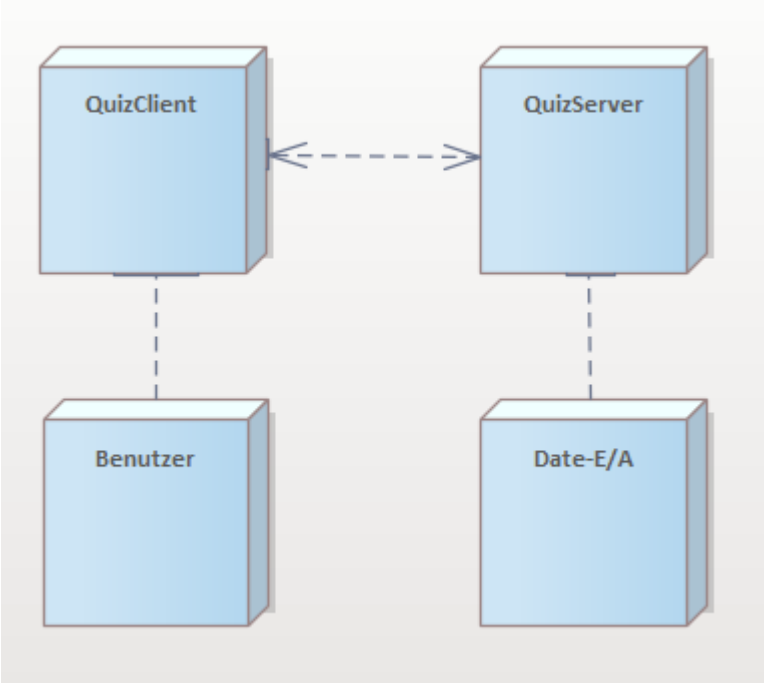
Der technische Kontext betrifft die technischen Aspekte des Systems, wie die verwendeten Technologien und die Art und Weise, wie die verschiedenen Teile des Systems zusammenarbeiten.

Java: Das gesamte System ist in Java programmiert, was es plattformunabhängig macht.

Netzwerk: Die Kommunikation zwischen dem Server und den Clients erfolgt über das Netzwerk mithilfe von TCP-Sockets.

Multithreading: Der Server verwendet Multithreading, um gleichzeitig mehrere Clients bedienen zu können.

Datei-E/A: Der Server liest die Quizdaten aus einer Textdatei.



In diesem Diagramm sendet und empfängt der QuizClient Daten vom Benutzer, während der QuizServer Daten von der Datei-E/A erhält. Die Kommunikation zwischen dem QuizClient und dem QuizServer erfolgt über das Netzwerk.

4 Lösungsstrategie

Die Lösungsstrategie unseres Quiz-Systems konzentriert sich auf Skalierbarkeit, Benutzerfreundlichkeit und Effizienz.

4.1 Skalierbarkeit

Das System ist so konzipiert, dass es mehrere Clients gleichzeitig unterstützen kann. Dies wird durch die Verwendung eines Multithreading-Ansatzes erreicht, bei dem jeder Client mit einem eigenen Thread bedient wird. Diese Strategie ermöglicht es dem System, mit der steigenden Anzahl von Benutzern umzugehen und eine konsistente Benutzererfahrung zu gewährleisten.

4.2 Benutzerfreundlichkeit

Das Quiz-System ist so gestaltet, dass es für die Benutzer einfach zu bedienen ist. Die Fragen und Antwortoptionen werden klar auf dem Bildschirm angezeigt und die Benutzer können ihre Antworten einfach durch Eingabe der entsprechenden Option (A, B oder C) wählen. Am Ende des Quiz wird der Gesamtpunktestand des Benutzers angezeigt.

4.3 Effizienz

Die Effizienz des Systems wird durch den effektiven Einsatz von Netzwerkressourcen und Datei-I/O-Operationen erreicht. Die Quizdaten werden einmalig beim Start des Servers von einer Textdatei gelesen und im Speicher gehalten, um die Anzahl der Datei-I/O-Operationen zu minimieren. Die Netzwerkkommunikation wird durch den Einsatz von TCP-Sockets optimiert, die eine zuverlässige Datenübertragung gewährleisten.

4.4 Technologieauswahl

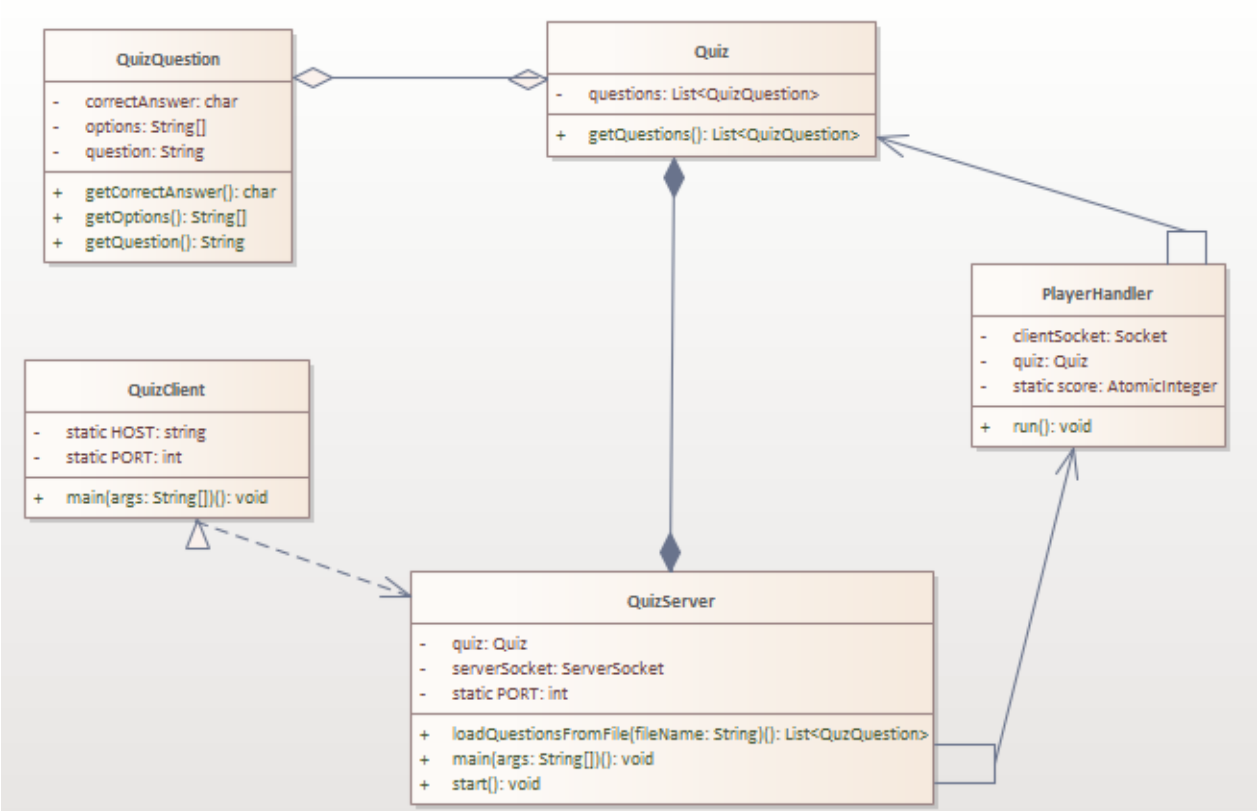
Wir haben uns für Java als Programmiersprache entschieden, weil es eine robuste Sprache mit eingebauter Unterstützung für Netzwerkkommunikation und Multithreading ist. Darüber hinaus ist Java plattformunabhängig, was bedeutet, dass unser Quiz-System auf verschiedenen Betriebssystemen laufen kann.

Zusammenfassend lässt sich sagen, dass unsere Lösungsstrategie auf Skalierbarkeit, Benutzerfreundlichkeit und Effizienz abzielt. Wir setzen bewährte Technologien und Praktiken ein, um ein robustes, benutzerfreundliches und effizientes Quiz-System zu entwickeln.

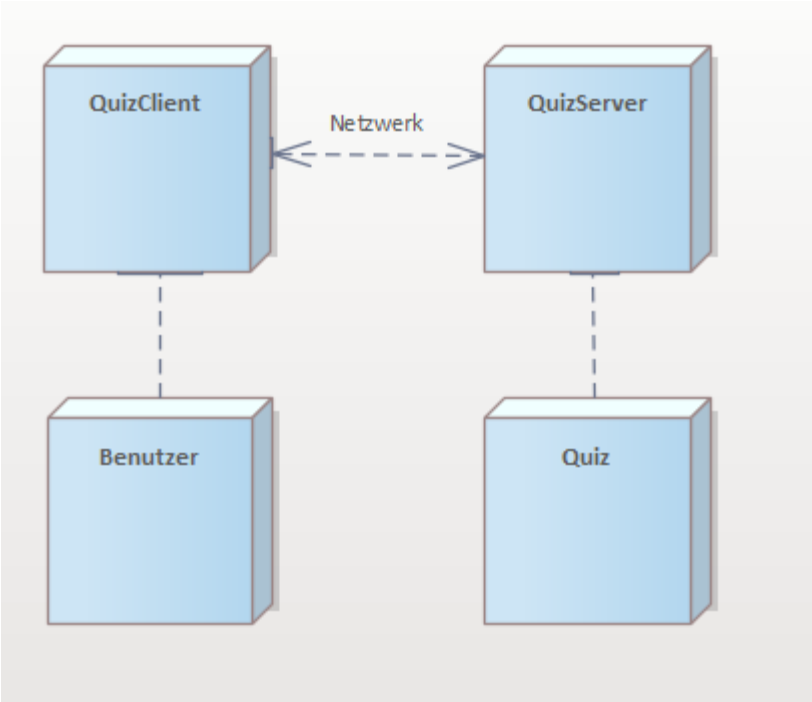
5. Bausteinsicht

Die Bausteinsicht zeigt die einzelnen Komponenten des Systems und wie sie zusammenarbeiten, um die Funktionalitäten des Systems bereitzustellen. Unser Quiz-System besteht aus vier Hauptkomponenten: QuizServer, PlayerHandler, QuizClient und Quiz.

UML Diagramm:



5.1 Level 1: Systemübersicht



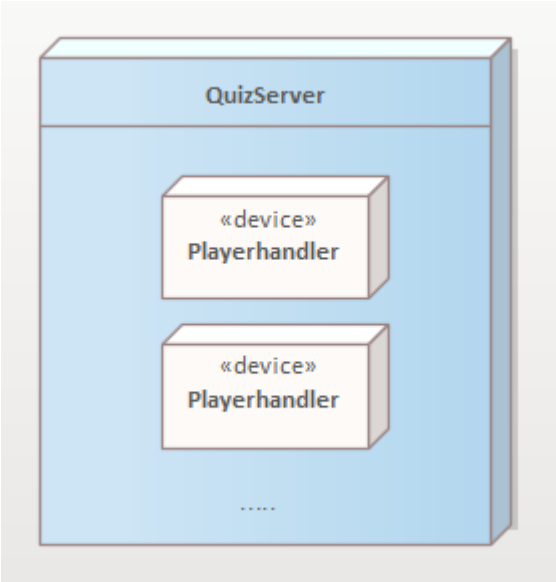
QuizClient: Der QuizClient ist die Schnittstelle, mit der der Benutzer interagiert. Er kommuniziert über das Netzwerk mit dem QuizServer, um Quizfragen zu erhalten und Antworten zu senden.

QuizServer: Der QuizServer verwaltet die Quizdaten und den Ablauf des Quiz. Er kommuniziert über das Netzwerk mit dem QuizClient, um Quizfragen zu senden und Antworten zu erhalten. Jeder QuizClient wird durch einen PlayerHandler bedient.

Quiz: Das Quiz besteht aus einer Reihe von Quizfragen. Es wird vom QuizServer verwaltet und an die QuizClients gesendet.

5.2 Level 2: QuizServer und PlayerHandler

Der QuizServer besteht aus mehreren PlayerHandlern, die jeweils einen QuizClient bedienen.

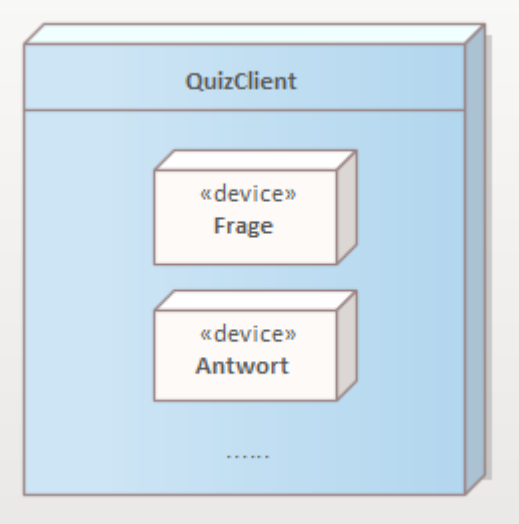


QuizServer: Der QuizServer startet die PlayerHandler-Threads und verwaltet das Quiz.

PlayerHandler: Jeder PlayerHandler bedient einen QuizClient. Er sendet Quizfragen an den QuizClient und erhält Antworten.

5.3 Level 3: QuizClient

Der QuizClient ermöglicht es dem Benutzer, das Quiz zu spielen. Eine detaillierte Ansicht des QuizClients könnte folgendermaßen aussehen:



Frage: Das QuizClient zeigt die Quizfrage an.

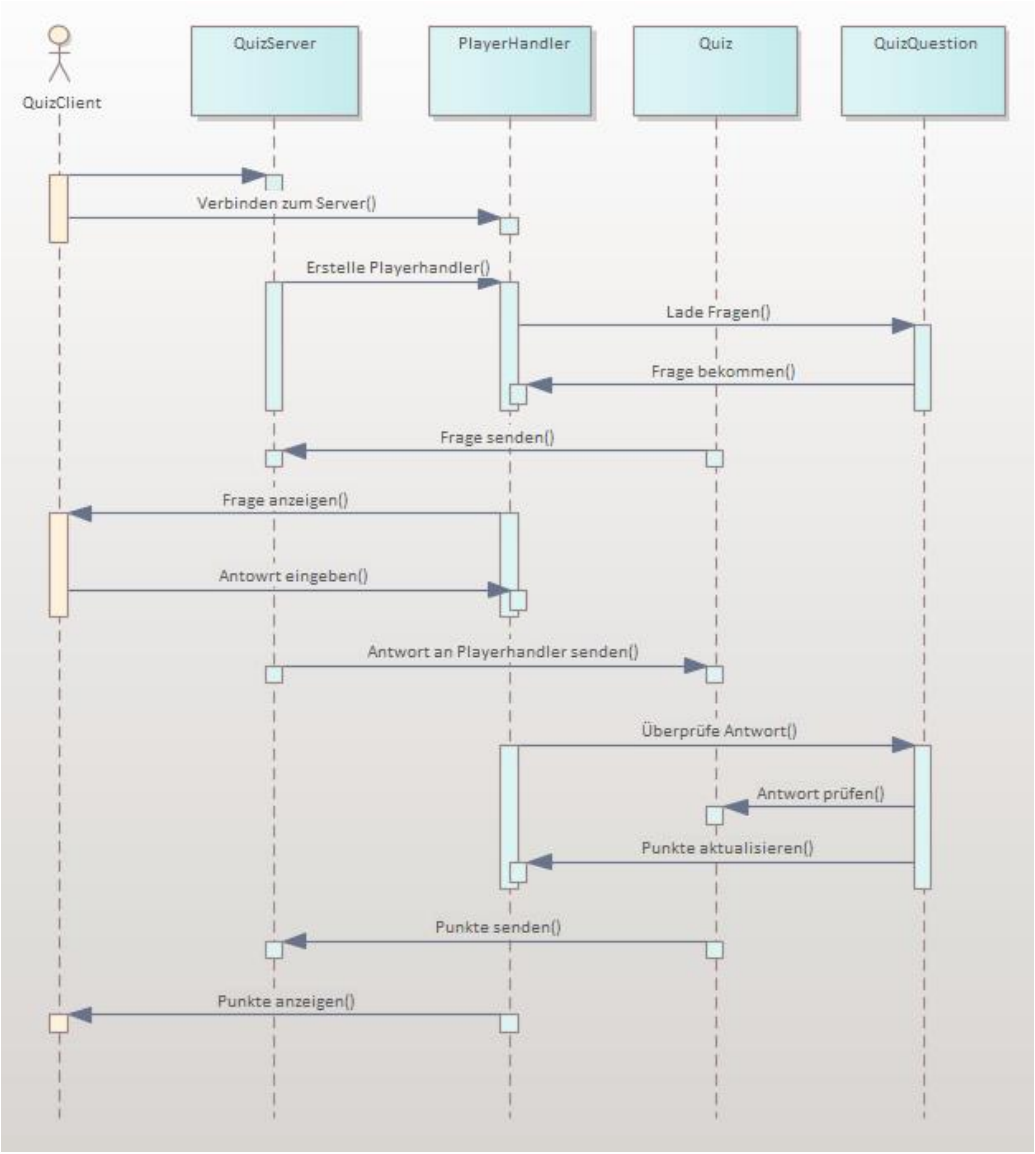
Antwort: Der Benutzer gibt die Antwort ein und das QuizClient sendet sie an den QuizServer.

6. Laufzeitsicht

Die Laufzeitsicht des Systems zeigt, wie die Hauptkomponenten während der Ausführung des Systems interagieren. In unserem Quiz-System sind die Hauptinteraktionen die Kommunikation zwischen dem QuizClient und dem QuizServer sowie die Abläufe innerhalb des QuizServers und des QuizClients.

6.1 Sequenzdiagramm

Ein Sequenzdiagramm kann dazu verwendet werden, um die Interaktionen zwischen den Komponenten während einer Quiz-Sitzung darzustellen:



In diesem Diagramm geht der QuizClient eine Verbindung mit dem QuizServer ein, der daraufhin einen PlayerHandler erstellt. Der PlayerHandler lädt die Fragen, sendet sie an den Client, erhält die Antworten, überprüft diese und aktualisiert schließlich die Punkte. Die Kommunikation findet größtenteils zwischen dem Client und dem Server statt, aber der Server interagiert auch mit den anderen Komponenten, um die Fragen zu laden und die Antworten zu überprüfen.

Verbindung zum Server: Der QuizClient initiiert eine Verbindung zum QuizServer indem er einen Socket auf dem gegebenen Host und Port erstellt.

Erstellen von PlayerHandler: Sobald der QuizServer eine Client-Verbindung akzeptiert hat, erstellt er einen neuen PlayerHandler, um die Kommunikation mit diesem spezifischen Client zu verwalten.

Fragen laden: Der PlayerHandler ruft die getQuestions Methode auf dem Quiz-Objekt auf, um eine Liste von QuizQuestion-Objekten zu erhalten. Diese repräsentieren die Fragen, die dem Spieler gestellt werden.

Frage senden: Der PlayerHandler sendet dann die erste Frage und ihre Optionen an den QuizClient über den Socket.

Frage anzeigen: Der QuizClient liest die Frage und die Optionen aus dem Socket und zeigt sie dem Benutzer an.

Antwort eingeben: Der Benutzer gibt seine Antwort ein, die vom QuizClient gelesen und dann an den QuizServer gesendet wird.

Antwort an PlayerHandler senden: Der QuizServer leitet die vom QuizClient gesendete Antwort an den entsprechenden PlayerHandler weiter.

Überprüfe Antwort: Der PlayerHandler ruft die Methode getCorrectAnswer auf dem aktuellen QuizQuestion-Objekt auf, um die korrekte Antwort zu erhalten, und vergleicht diese mit der Antwort des Spielers.

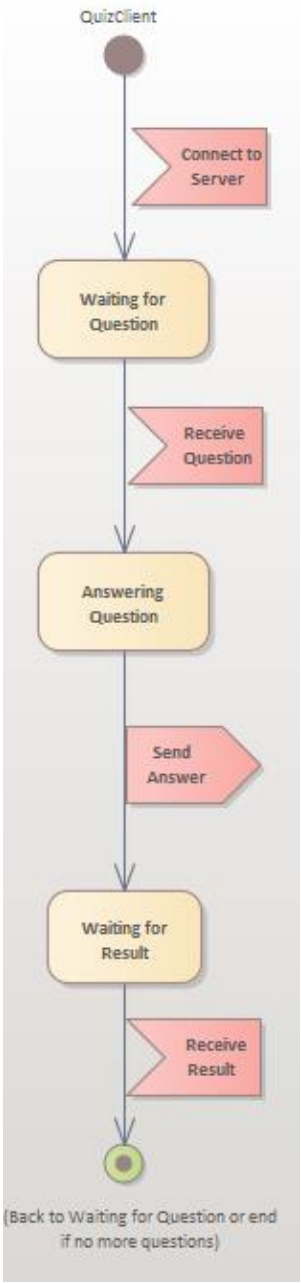
Punkte aktualisieren: Abhängig vom Ergebnis des vorherigen Schrittes aktualisiert der PlayerHandler die Punktzahl des Spielers.

Punkte senden: Der PlayerHandler sendet dann die aktualisierte Punktzahl an den QuizClient über den Socket.

Punkte anzeigen: Der QuizClient liest die Punktzahl aus dem Socket und zeigt sie dem Benutzer an.

Dieser Prozess wird für jede Frage in der Quizfragenliste wiederholt. Am Ende des Quiz wird eine Endnachricht an den Client gesendet und die Verbindung wird geschlossen.

6.2 Zustandsdiagramm



Connect to Server: Der QuizClient stellt eine Verbindung zum QuizServer her und geht in den Zustand "Waiting for Question" über.

Waiting for Question: Der QuizClient wartet auf eine Frage vom QuizServer.

Receive Question: Der QuizClient empfängt eine Frage und geht in den Zustand "Answering Question" über.

Answering Question: Der QuizClient ermöglicht es dem Benutzer, die Frage zu beantworten.

Send Answer: Der QuizClient sendet die Antwort an den QuizServer und geht in den Zustand "Waiting for Result" über.

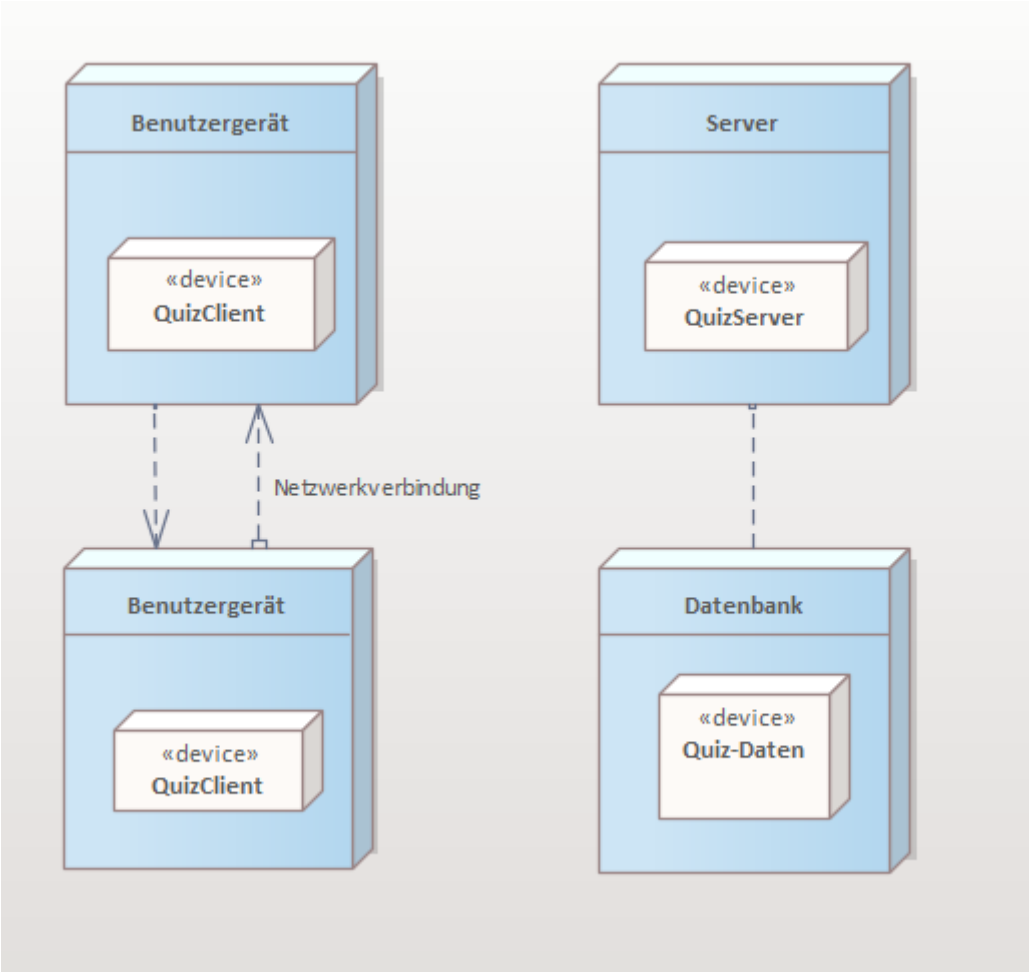
Waiting for Result: Der QuizClient wartet auf das Ergebnis vom QuizServer.

Receive Result: Der QuizClient empfängt das Ergebnis und geht zurück in den Zustand "Waiting for Question", wenn es noch weitere Fragen gibt, oder beendet das Quiz, wenn es keine weiteren Fragen gibt.

7. Verteilungssicht

Die Verteilungssicht beschreibt, wie das System auf verschiedene physische Ressourcen verteilt ist. Unser Quiz-System ist ein Client-Server-System, bei dem die Serverkomponenten auf einem Server laufen und die Clientkomponenten auf den Geräten der Benutzer.

7.1 Systemübersicht



Benutzergerät: Die Benutzergeräte enthalten die QuizClient-Anwendung, die es dem Benutzer ermöglicht, das Quiz zu spielen. Jedes Gerät kommuniziert über das Netzwerk mit dem Server.

Server: Der Server beherbergt die QuizServer-Anwendung, die die Quiz-Daten verwaltet und die Kommunikation zwischen den QuizClients ermöglicht. Er kommuniziert auch mit der Datenbank, um Quiz-Daten zu lesen und zu schreiben.

Datenbank: Die Datenbank enthält die Quiz-Daten, einschließlich der Fragen, Antworten und möglicherweise der Benutzerstatistiken. Sie wird vom Server gelesen und geschrieben.

Die tatsächliche Verteilung des Systems hängt von vielen Faktoren ab, einschließlich der Anzahl der Benutzer, der erforderlichen Netzwerkbandbreite, der Leistung des Servers und der Datenbank, und so weiter. Es könnte notwendig sein, Teile des Systems auf mehrere Server zu verteilen oder mehrere Datenbankinstanzen zu verwenden, um die Leistung zu verbessern oder die Zuverlässigkeit zu erhöhen.

8 Querschnittliche Konzepte

In dieser Sektion werden die grundlegenden Konzepte des Systems und ihre Beziehungen erläutert. Im Kontext unseres Quiz-Systems könnten die wichtigsten Konzepte die folgenden sein:

Quiz: Ein Quiz ist eine Sammlung von Fragen, die den Benutzern gestellt werden. Jede Frage hat mehrere Antwortmöglichkeiten, von denen eine korrekt ist.

QuizServer: Der QuizServer ist die Hauptkomponente des Systems, die das Quiz verwaltet. Er liest die Fragen aus der Datenbank, sendet sie an die QuizClients und überprüft die Antworten der Benutzer.

QuizClient: Der QuizClient ist die Komponente, die auf den Geräten der Benutzer läuft. Er empfängt die Fragen vom QuizServer, zeigt sie dem Benutzer an und sendet die Antworten des Benutzers an den Server.

PlayerHandler: Der PlayerHandler ist eine Komponente des QuizServers, die die Kommunikation mit einem bestimmten QuizClient verwaltet. Es gibt einen PlayerHandler für jeden verbundenen QuizClient.

QuizQuestion: Eine QuizQuestion repräsentiert eine einzelne Frage im Quiz. Sie enthält den Text der Frage, die möglichen Antworten und die korrekte Antwort.

Diese Konzepte bilden die Grundlage für die Funktionalität des Systems und bestimmen, wie die verschiedenen Komponenten interagieren und wie die Daten im System organisiert sind.

9 Architekturentscheidungen

In diesem Abschnitt werden die wichtigsten Entwurfsentscheidungen dokumentiert, die während der Entwicklung des Systems getroffen wurden. Für unser Quiz-System könnten das folgende Entscheidungen sein:

Wahl des Client-Server-Modells: Wir haben uns für ein Client-Server-Modell entschieden, da es uns ermöglicht, die Quiz-Daten zentral zu verwalten und gleichzeitig viele Benutzer zu unterstützen. Außerdem können wir so sicherstellen, dass die Antworten der Benutzer korrekt und fair bewertet werden.

Verwendung von Java: Wir haben uns für Java entschieden, weil es eine robuste und weit verbreitete Sprache ist, die viele Funktionen für Netzwerkkommunikation und Multithreading bietet.

Multithreading im Server: Wir haben uns dafür entschieden, jeden QuizClient in einem eigenen Thread zu bedienen, um eine hohe Leistung zu erreichen und viele gleichzeitige Benutzer zu unterstützen.

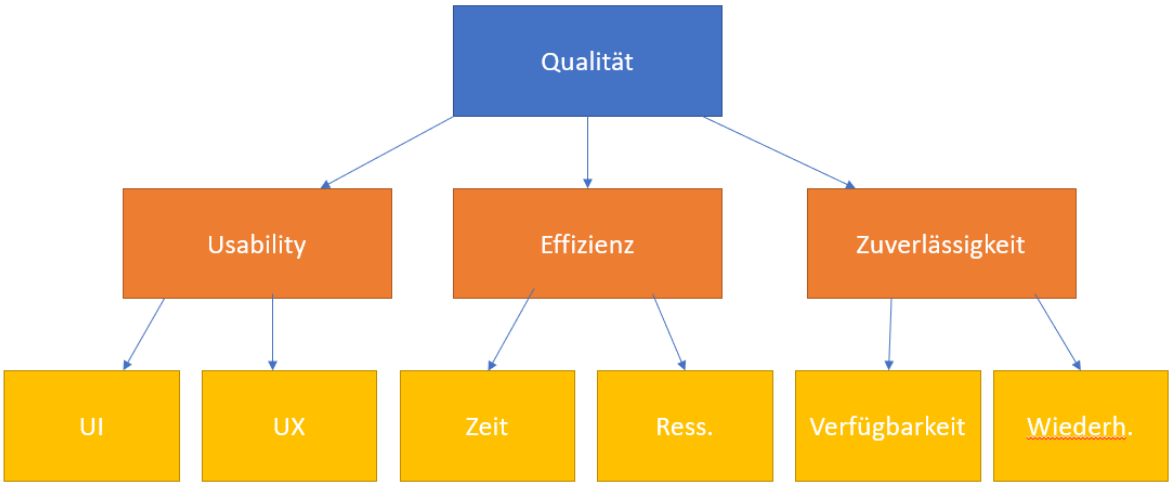
Textbasiertes Protokoll: Wir haben uns für ein einfaches textbasiertes Protokoll entschieden, um die Kommunikation zwischen Client und Server zu vereinfachen und die Entwicklung zu beschleunigen.

Diese und andere Entwurfsentscheidungen haben einen erheblichen Einfluss auf die Struktur und Funktionalität des Systems. Es ist wichtig, diese Entscheidungen zu dokumentieren, damit sie bei zukünftigen Änderungen oder Erweiterungen des Systems berücksichtigt werden können.

10 Qualitätsanforderungen

Qualitätsszenarien beschreiben, wie das System unter bestimmten Bedingungen reagiert, um die Qualität des Systems zu bewerten. Für unser Quiz-System könnten Qualitätsszenarien beispielsweise die folgenden Aspekte abdecken:

10.1 Qualitätsbaum



Usability (Benutzbarkeit): Wie einfach und intuitiv ist das System zu bedienen?

UI (User Interface): Bezieht sich auf das Design der Benutzeroberfläche und wie benutzerfreundlich sie ist.

UX (User Experience): Bezieht sich auf die Gesamterfahrung, die ein Benutzer hat, wenn er das System verwendet.

Effizienz: Wie effizient ist das System bei der Verwendung seiner Ressourcen?

Zeit: Wie schnell reagiert das System auf Benutzeranfragen und führt Operationen aus?

Ressourcen: Wie effizient nutzt das System seine Ressourcen, wie Speicher und Prozessorleistung?

Zuverlässigkeit: Wie zuverlässig und stabil ist das System?

Verfügbarkeit: Wie oft ist das System verfügbar und bereit, Anfragen zu bearbeiten?

Wiederherstellbarkeit: Wie gut kann das System nach einem Fehler wiederhergestellt werden und in einen funktionsfähigen Zustand zurückkehren?

Jede dieser Qualitäten kann weiter in spezifischere Unterkategorien unterteilt werden, je nach den spezifischen Anforderungen des Systems. Bitte beachten Sie, dass dies nur ein Beispiel für einen Qualitätsbaum ist, und der tatsächliche Qualitätsbaum kann je nach den spezifischen Anforderungen und Zielen des Projekts variieren.

10.2 Qualitätsszenarien

Leistung: Wie viele Benutzer kann das System gleichzeitig bedienen? Wie schnell reagiert das System auf Anfragen von Benutzern? Wie wirkt sich die Netzwerklatenz auf die Leistung aus?

Zuverlässigkeit: Wie robust ist das System gegenüber Netzwerkfehlern oder Ausfällen von Komponenten? Was passiert, wenn ein Benutzer während eines Quiz die Verbindung verliert?

Sicherheit: Wie sicher ist das System gegenüber Manipulationsversuchen? Können Benutzer die Antworten auf die Fragen im Voraus sehen oder ihre Punktzahl manipulieren?

Benutzerfreundlichkeit: Wie einfach ist es für Benutzer, das Quiz zu spielen? Wie klar sind die Fragen und Antworten? Wie gut funktioniert das Benutzerinterface auf verschiedenen Geräten und Bildschirmgrößen?

Diese Szenarien können dazu verwendet werden, um Tests zu entwerfen, um die Qualität des Systems zu bewerten, oder um Anforderungen für die Entwicklung neuer Funktionen zu definieren.

11 Risiken und technische Schulden

In diesem Abschnitt werden mögliche Risiken und technische Schulden, die im Laufe der Entwicklung entstanden sind, dokumentiert. Diese können sowohl aus technischen als auch aus organisatorischen Gründen entstehen. Einige mögliche Punkte für unser Quiz-System könnten beispielsweise sein:

Skalierung: Da der Server alle Anfragen verarbeitet und alle Benutzer gleichzeitig bedient, könnte die Skalierbarkeit bei sehr vielen Benutzern zu einem Problem werden. Es könnte sinnvoll sein, in Zukunft eine Lastverteilung zu implementieren.

Fehlerbehandlung: Momentan ist die Fehlerbehandlung sehr einfach gestaltet und es gibt wenig Rückmeldung an den Benutzer. Ein technischer Schuldenpunkt könnte sein, diese Aspekte zu verbessern und ein ausgereiftes Fehlerbehandlungs- und Logging-System zu implementieren.

Sicherheit: Derzeit gibt es keine Authentifizierung der Benutzer, was zu Sicherheitsproblemen führen könnte. Es könnte sinnvoll sein, eine Authentifizierung zu implementieren, um sicherzustellen, dass nur autorisierte Benutzer auf das Quiz zugreifen können.

12 Glossar

Begriff	Definition
Quiz	Ein Test, bei dem Benutzer Fragen beantworten und Punkte sammeln können.
QuizServer	Die Anwendung, die das Quiz verwaltet, die Fragen sendet und die Antworten der Benutzer überprüft.
PlayerHandler	Eine Komponente des QuizServers, die die Kommunikation mit einem bestimmten QuizClient verwaltet.
QuizQuestion	Ein Objekt, das eine einzelne Frage im Quiz, die möglichen Antworten und die korrekte Antwort repräsentiert.
QuizClient	Die Anwendung, die auf den Geräten der Benutzer läuft und es ihnen ermöglicht, das Quiz zu spielen.