Architekturbeschreibung: C#-basierte Notenverwaltungs-Desktop-App

1. Überblick

Diese Dokumentation beschreibt die Architektur einer C#-basierten Client-Server-Desktop-Anwendung zur Verwaltung von Noten. Die Anwendung bietet eine intuitive Benutzeroberfläche für Lehrkräfte und Schüler, die das Erstellen und Bearbeiten von Noten ermöglicht. Sie unterstützt zwei Modi: einen Offline-Modus, der die Arbeit ohne Internetverbindung erlaubt, sowie einen Online-Modus, der die Synchronisierung zwischen lokalen und Remote-Datenbanken gewährleistet.

2. Komponenten

a) Benutzeroberfläche (UI)

Die Benutzeroberfläche ist als Konsolenanwendung umgesetzt, um eine schnelle Entwicklung zu ermöglichen. Sie erlaubt die Navigation und Auswahl von Optionen über Tastaturbefehle, insbesondere Pfeiltasten.
Zur Strukturierung der Anwendung wird das MVC-Architekturmuster (Model-View-Controller) eingesetzt, um die Trennung von Präsentation, Steuerung und Datenlogik zu gewährleisten.

b) Logik

1. Online-Betrieb:
- Der Client sendet HTTP-Anfragen an den Server.
- Die serverseitige Logik verarbeitet diese Anfragen und sendet Daten zurück an den Client, und an die Datenbank.

2. Offline-Betrieb:
- Der Client nutzt lokale Logik, um Änderungen an einer SQLite-Datenbank vorzunehmen.
Eine Abfrage prüft den aktuellen Modus der Anwendung und entscheidet, ob

Aktionen lokal oder serverseitig ausgeführt werden.

3. Synchronisierung

-Die Logik zur Synchronisierung der Datenbanken wird Client-Seitig ausgeführt, kann

jedoch in späteren Iterationen, des Projekts in Microservices ausgelagert werden.

c) Datenbank

Die Anwendung verwendet zwei Datenbanken:
- Lokale SQLite-Datenbank: Speichert Daten offline und dient als Synchronisationsziel. - Remote SQLite/MySQL-Datenbank: Wird im Online-Modus verwendet und läuft zu Testzwecken auf demselben Gerät wie der Client.

Im Online-Modus werden Änderungen parallel in beiden Datenbanken gespeichert, um eine nahtlose Synchronisierung zu gewährleisten. Zur Verwendung der Datenbank werden die System.Data.SQLite / MySql.Data.MySqlClient Bibliotheken benutzt.

3. Kommunikationsmechanismen

Die Kommunikation zwischen Client und Server erfolgt über HTTP-APIs.

- Client: Sendet Anfragen und interpretiert die serverseitigen Antworten.
- Server: Verarbeitet eingehende Anfragen und gibt die Ergebnisse zurück.

Daten werden als JSON versendet, und zur Verarbeitung von JSON-Daten wird die Bibliothek System.Text.Json verwendet. Diese sorgt für effiziente Serialisierung und Deserialisierung der Nachrichten.

4. Sicherheitsaspekte

a) Authentifizierung

Da die Anwendung keine Benutzerverwaltung enthält, müssen Nutzer vorab manuell in der Datenbank angelegt werden.

- Erstanmeldung: Erfolgt online, um Benutzername und Passwort gegen die Serverdatenbank zu verifizieren.
- JWT-Token: Nach erfolgreicher Verifizierung wird ein JSON Web Token (JWT) lokal gespeichert. Dieses Token ermöglicht die Offline-Anmeldung ohne erneute Serververbindung.

Verwendete Bibliotheken:
- Microsoft.IdentityModel.Tokens
- System.IdentityModel.Tokens.Jwt
- System.Security.Claims

b) Datensicherheit bei der Übertragung

Zur Sicherstellung der Datenintegrität:

Verwendet der Client Repository-Muster, um empfangene JSON-Daten vor der Verarbeitung zu validieren.

Serverseitig werden alle eingehenden Daten ebenfalls geprüft, um unerwartete oder fehlerhafte Formate abzuwehren.

c) Validierung der Benutzereingaben

Die Validierung von Benutzereingaben ist aus Zeitgründen noch nicht implementiert. Zusätzliche Sicherheitsmaßnahmen wie Anfrage-Limits oder Schutzmechanismen gegen Missbrauch sind aktuell ebenfalls nicht vorgesehen.

5. Skalierbarkeit und Erweiterbarkeit

Die Architektur ist modular aufgebaut, was zukünftige Erweiterungen erleichtert.
Zum Beispiel:

- Benutzerverwaltung: Neue Funktionen wie die Registrierung und Verwaltung von Benutzern können durch Hinzufügen entsprechender Controller und Schnittstellen implementiert werden.

- Funktionserweiterungen: Sowohl auf Client- als auch auf Serverseite können neue Features durch Ergänzung der bestehenden Module integriert werden.