TWMailer Basic Protocol

**1. Client und Server Architektur**

Die entwickelte Anwendung besteht aus einem Client und einem Server, die auf der Basis des TCP/IP-Protokolls über eine Netzwerkverbindung miteinander kommunizieren. Die grundlegende Architektur dieser Anwendung folgt einem klassischen **Client-Server-Modell**, bei dem der Server als zentraler Nachrichtenspeicher und -verwalter agiert, während der Client verschiedene Befehle zum Senden, Empfangen und Löschen von Nachrichten an den Server schickt.

* **Client:**
  + Der Client stellt die Benutzerschnittstelle bereit und ermöglicht dem Benutzer das Eingeben von Befehlen wie SEND, LIST, READ, DEL, und QUIT.
  + Nach der Eingabe eines Befehls formatiert der Client die Eingaben und sendet diese über eine TCP-Verbindung an den Server. Die Datenübertragung erfolgt im Textformat, sodass der Server die Befehle verarbeiten kann.
  + Für die Implementierung der Netzwerkkommunikation wird das POSIX Socket-API genutzt, was in der Funktionalität der Socket-Erstellung, Verbindung und Datenübertragung sichtbar ist.
* **Server:**
  + Der Server empfängt die vom Client gesendeten Befehle, verarbeitet diese und sendet die entsprechenden Antworten zurück.
  + Der Server speichert alle Nachrichten in einem lokalen Verzeichnis, wobei für jeden Benutzer ein eigenes Verzeichnis angelegt wird. Jede Nachricht wird in einer separaten Datei gespeichert.
  + Der Server ist in der Lage, mehrere Clients zu bedienen. Für jede eingehende Client-Verbindung wird ein neuer Socket verwendet.
  + Der Server verwendet das Dateisystem, um Nachrichten zu speichern, indem die Bibliothek std::filesystem verwendet wird. Jede Nachricht wird als Textdatei im jeweiligen Verzeichnis des Benutzers abgelegt.

**2. Verwendete Technologien**

Die Haupttechnologien, die für die Entwicklung dieser Client-Server-Anwendung verwendet wurden, umfassen:

* **Programmiersprache:**
  + Das Programm wurde vollständig in **C++** geschrieben und verwendet den C++17-Standard, um moderne Sprachfeatures zu nutzen.
* **Netzwerkkommunikation:**
  + Für die Implementierung der Netzwerkkommunikation wurde die **POSIX-Socket-API** verwendet. Diese ermöglicht die Erstellung von TCP/IP-Sockets sowohl auf der Client- als auch auf der Serverseite.
  + Der Client erstellt eine TCP-Verbindung zum Server über die Funktionen socket(), connect() und send(), während der Server socket(), bind(), listen() und accept() verwendet, um auf eingehende Verbindungen zu warten.
* **Dateisystem:**
  + Der Server nutzt die std::filesystem-Bibliothek, um Nachrichten im lokalen Dateisystem zu speichern. Diese moderne C++-Bibliothek bietet eine bequeme Möglichkeit, Verzeichnisse zu erstellen, zu überprüfen und Dateien zu verwalten.
* **String-Manipulation:**
  + Sowohl der Client als auch der Server verwenden Stringstreams (std::stringstream und std::ostringstream), um die Nachrichteninhalte zu verarbeiten und die Kommunikation zu formatieren.

**3. Entwicklungsstrategie und notwendige Anpassungen**

Die Entwicklungsstrategie folgte einem **iterativen Ansatz**, wobei der Fokus zunächst auf der Implementierung der Grundfunktionen wie SEND, LIST, READ und DEL lag. Zu den Schritten gehörten:

* **Schrittweise Implementierung der Client-Funktionalitäten:**
  + Zunächst wurden die grundlegenden Funktionen auf der Client-Seite implementiert, angefangen mit der SEND-Funktion. Sobald der Client in der Lage war, Nachrichten korrekt an den Server zu senden, wurden die weiteren Funktionen wie LIST (Auflisten der Nachrichten), READ (Lesen einer Nachricht) und DEL (Löschen einer Nachricht) hinzugefügt.
* **Serverentwicklung:**
  + Parallel zur Client-Entwicklung wurde die Serverlogik entwickelt. Die Hauptaufgabe des Servers bestand darin, die vom Client empfangenen Befehle zu interpretieren und entsprechend zu handeln (z.B. Nachrichten im Dateisystem zu speichern oder zurückzugeben).
  + Eine wesentliche Herausforderung bestand darin, eine konsistente Speicherung und Verwaltung der Nachrichten auf dem Server zu gewährleisten. Hierbei kam die std::filesystem-Bibliothek ins Spiel, die effiziente Datei- und Verzeichnisoperationen ermöglichte.
* **Anpassungen und Verbesserungen:**
  + **Fehlerbehandlung:** Während der Entwicklung wurden Fehlerbehandlungsmechanismen hinzugefügt, um sicherzustellen, dass Verbindungsfehler, falsche Eingaben oder fehlende Dateien korrekt gehandhabt werden. Beispielsweise wird überprüft, ob ein Benutzerverzeichnis existiert, bevor Nachrichten aufgelistet oder gelesen werden.
  + **Speicherverwaltung:** Um sicherzustellen, dass der Server effizient arbeitet, wurden Maßnahmen ergriffen, um den Speicherverbrauch zu minimieren, wie etwa das gezielte Leeren von Puffern nach jeder Nachricht.

**4. Zusammenfassung**

Die TW-Mailer-Anwendung verwendet eine einfache und robuste **Client-Server-Architektur**, die durch die Nutzung von POSIX-Sockets und der modernen C++-Bibliotheken unterstützt wird. Die iterative Entwicklungsstrategie ermöglichte es, eine solide Grundlage zu schaffen, auf der zusätzliche Funktionen problemlos implementiert werden konnten. Anpassungen während der Entwicklung verbesserten die Fehlerbehandlung und die Speicherverwaltung, um eine stabile und performante Anwendung bereitzustellen.